07-17-00 A

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	065489.00001
First Inventor or Application Identifier	Evan E. Dussia
Title	Computerized Method And System For.....
Express Mail Label No.	EK552366385US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

1. [X] * Fee Transmittal Form *(e.g., PTO/SB/17)* *(Submit an original and a duplicate for fee processing)*
2. [X] Specification [Total Pages 24] *(preferred arrangement set forth below)*
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [X] Drawing(s) *(35 U.S.C. 113)* [Total Sheets 4]
4. Oath or Declaration [Total Pages]
   a. [ ] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

* NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. [ ] Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [ ] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
9. [ ] English Translation Document *(if applicable)*
10. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. [ ] * Small Entity Statement(s) *(PTO/SB/09-12)* [ ] Statement filed in prior application, Status still proper and desired
14. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. [X] Other: Express Mailing Certificate

16. **If a CONTINUING APPLICATION,** *check appropriate box, and supply the requisite information below and in a preliminary amendment:*

[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: _____/_____

Prior application information: Examiner _____ *Group / Art Unit:* _____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [X] *Correspondence address below*

Name	James H. Beusse, Esquire				
	Holland & Knight, LLP				
Address	P.O. Box 1526				
City	Orlando	State	FL	Zip Code	32802-1526
Country	U.S.	Telephone	(407) 244-1129	Fax	(407) 244-5288

Name (Print/Type)	James H. Beusse	Registration No. (Attorney/Agent)	27,115
Signature		Date	07/14/2000

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

VIA EXPRESS MAIL EK552366385US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of		)
		)
Applicant:	Evan E. Dussia	)
		)
Application No.:	Not Assigned	)
		)
Filing Date:	Simultaneously Herewith	)
		)
For:	Computerized Method And System For Obtaining, Storing And Accessing Medical Records	)

Commissioner of Patents
Box Patent Application
Washington, D.C. 20231

Sir:

EXPRESS MAIL # EK552366385US

Date of Deposit: July 14, 2000

I hereby certify that the attached patent application, formal papers and informal drawings were deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Commissioner of Patents, Box Patent Application, Washington, D.C. 20231.

Respectfully submitted,

Bonnie A. Brennan

Bonnie A. Brennan
Holland & Knight LLP
P.O. Box 1526
Orlando, FL 32802-1526
(407) 244-1122

COMPUTERIZED METHOD AND SYSTEM FOR OBTAINING, STORING AND ACCESSING MEDICAL RECORDS

This application claims the benefit of U.S. provisional application No. 60/185,577 filed on February 28, 2000.

BACKGROUND OF THE INVENTION

The present invention relates to a system and method for managing medical patient records and, more particularly, to a computerized system and method which provides for the capture, storage, processing, communication, security and presentation of non-redundant patient health information over an Internet connection.

It is believed that prior to the present invention nearly all communication between doctors' offices regarding patient records has been by paper or by telephone. This is also true of communications between doctors' offices and insurers, HMOs, MCOs, hospitals, and pharmacies. Known medical data systems have suffered from undesirable data cluttering due to attempting to be all encompassing and have been generally designed around information gathering parameters, rather than providing narrowly focused and unobtrusive management of the patient records. It is further believed that no single system has gained wide acceptance for medical records management in the office environment, and none has been designed for data sharing among multiple users.

In view of the above-described issues, it would be desirable to provide a system that is user-friendly, and provides straightforward inquiry screens which display essential patient information from the doctor's viewpoint, such as diagnosis and treatment plans. It would be further desirable to enable the user "to see" into the thought process of the treating physician. If more detailed information is desired, the viewer may just "point and click" to see the entire text of the physician's progress (or encounter) notes while avoiding the potential for data cluttering due to useful but not necessarily essential data, such as lab tests, scans, and X-rays images.

It is also desirable to provide a system and method that would:

- Assist specialist practitioners to whom a patient has been referred, by eliminating most requests for file data from referring physicians, and removing guesswork and office time involved in obtaining a complete list of medications which the patient may be using.

- Make after hours and emergency hospital visits more risk free because of the availability of patient information on a permanent round-the-clock basis. At present, records are generally unavailable when the physicians' offices are closed. Additionally, records are fragmented and scattered among all practitioners with whom the patient is associated. Even in those rare instances where the patient in the emergency room is knowledgeable concerning his own medical history and drug therapies, if he or she is in shock, in great pain, or unconscious, it is currently difficult for the attending physician to quickly obtain patient medical data.

- Enable druggists to avoid drug interactions and allergic reactions in filing prescriptions. Since patients may obtain their prescription medications from more than one location, then it is currently very difficult, if not impossible, for pharmacists to fulfill their potential in helping patients avoid adverse drug interactions.

- Enable health insurers, such as HMOs and managed care companies, to perform the required quality assurance inspections and utilization reviews off-site with the click of a button at a fraction of the current costs. Both of these functions are currently performed by inspectors and auditors who actually go to the practitioner's office and have his/her staff pull files. The individual files are then reviewed and certain documents (i.e., the progress notes, problem list, treatment plan and drug list) may be copied by the doctor's staff. Then the files need to be returned to their proper place and annotated.

This process is expensive for both companies and for the physicians, which of course translates into higher costs for patients.

SUMMARY OF THE INVENTION

Generally speaking, the present invention in one aspect thereof fulfills the foregoing needs by providing a medical health record storage and retrieval system comprising an interface module configurable to extract a patient's medical diagnosis and treatment from respective progress notes of a physician. A storage module is configured to store the extracted diagnosis and treatment in a logically connected database. A server is configured to allow access to the stored database by authorized users, and a processor module is configured to track users accessing the database and to bill the accessing users for each access of the database.

The present invention further fulfills the foregoing needs by providing in another aspect thereof a computerized method for managing respective health records of a plurality of patients. The method allows for uploading a progress note of a respective patient. The progress note comprises data relative to an encounter between a respective physician and the respective patient. The method further allows for identifying on the progress note respective parameters selectable by the respective physician. A storing step allows for storing the progress note with the identified parameters in a database accessible to a plurality of authorized users. A populating step allows for populating the database with respective progress notes resulting from further encounters between the respective patient and any respective physician so as to create a historical set of progress notes for that respective patient.

In yet another aspect thereof, the present invention further provides a computer-readable medium encoded with computer program code for managing respective health records of a plurality of patients. The program code causes a computer to execute a method comprising:

uploading a progress note of a respective patient, said progress note comprising data relative to an encounter between a respective physician and the respective patient;

identifying on said progress note respective parameters selectable by the respective physician;

storing said progress note with said identified parameters in a database accessible to a plurality of authorized users; and

populating said database with respective progress notes resulting from further encounters between the respective patient and any respective physician so as to create a historical set of

progress notes for that respective patient, the set of historical progress notes being interconnectable based on one or more logic operators.

DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an exemplary block diagram of a system for managing medical records that embodies one aspect of the present invention;

FIG. 2 illustrates an exemplary Web page comprising a progress report based on a respective encounter between a patient and physician including respective diagnosis and prescription data extracted by that physician;

FIG. 3 illustrates an exemplary Web page comprising historical data of diagnosis and prescriptions based on respective progress reports such as shown in FIG. 3; and

FIG. 4 illustrates an exemplary layout of a data structure including respective data fields for managing medical records.

Before any embodiment of the invention is explained in detail, it is to be understood that the invention is not limited in its application to the details of construction and the arrangements of components set forth in the following description or illustrated in the drawings. The invention is capable of other embodiments and of being practiced or being carried out in various ways. Also, it is to be understood that the phraseology and terminology used herein is for the purpose of description and should not be regarded as limiting.

DETAILED DESCRIPTION OF THE INVENTION

Definitions, Acronyms and Abbreviations

Physician: A user representing a physician or physician's office, who does not necessarily has to be a physician but has access to the system.

Pharmacy: A user representing a pharmacy or a pharmacist, having access to the system.

Hospital: A user representing a hospital, having access to the system.

Insurance Company: A user representing an insurance company, having access to the system.

Pharmaceutical Company: A user representing a pharmaceutical company, having access to the system.

Transcription Service/Transcriber: Transcription services are users of the system responsible for entering progress notes into the system.

User: Anyone with an account allowing them access to the system.

GUI: Graphical User Interface. A graphics-based interface using icons, pull down menus, and other computer peripherals, e.g., a mouse, that enables the user to interact with the system.

HTML: HyperText Markup Language. The scripting language used to build web pages.

ASP: Active Server Pages. A technique for using programming and logic to dynamically process web pages on a server, as they are requested from a client browser. ASP pages will be able to interface with COM objects running in a transaction server, allowing the application access to the data through a business layer contained in a COM object.

COM: Component Object Model. A model for developing architectural components that allows for configuring a component so that other applications and programs may interface with it.

IIS: Internet Information Server. A Web server available from Microsoft that allows client browsers to request web pages and serves those pages to the browsers. The user interface to the application will comprise web pages being served and maintained by IIS.

Transaction Server: A server, also available from Microsoft, that manages the COM objects. A middle-tier business logic of the application will be contained within a COM component running within the transaction server.

MS SQL: Microsoft database server based on structured query language. SQL will store the data tables, views, triggers and stored procedures that process the data. SQL is accessed from the application through the COM object running on Transaction Server.

Progress Note: A progress note is a description of the physician's meeting with a patient. The information contained within a progress note comprises the time and date of the note, the physician, the patient, and a description of the reason for the visit. This description will generally include the diagnosis or diagnoses and any necessary prescription(s).

Diagnosis: A physician's diagnosis.

Prescription: A prescription prescribed for a diagnosis.

Owner: The physician who is responsible for entering the patient into the system. This will not necessarily be the patient's primary physician.

Parent: When a new physician is referred to the system and becomes a participant or subscriber, the physician who referred the new physician is the new physician's parent.

CSS: Cascading Style Sheets. CSS 1.0 is a specification for the authoring of style sheets within an HTML document. The style sheets assist in code reuse and in giving web pages a similar look and feel across an application.

SSL: Secure Socket Layer. A technique for sending encrypted data across public data lines.

Billing/Maintenance Information: This term labels an entity's address and contact information. This is to distinguish between patients and users of the system. Patient's information will be referred to as demographic information and will include address and contact information, as well as insurance information, if available. Other system entities' address and contact information will be referred to as Billing/Maintenance Information.

Insert: Insert permission allows the user the ability to add records to the system. If the user does not have insert permission for an entity type, they will not be able add new records of that entity type.

Update: Update permission allows the user the ability to update existing records in the system. If the user does not have update permission for an entity type, they will not be able update records of that entity type.

Overview

The present invention in one aspect thereof permits universal and timely access to health information for approved care givers and other authorized users 24 hours a day, seven days a week using a web-enabled system 10, as shown in FIG. 1. The system allows on-line access to a patient's health history by approved caregivers. The system uses the physician's work product in the form of dictated progress notes as the foundation document for extracting a patient's medical history. An extraction module operates in a free text environment of the dictated progress note to extract the problem or diagnosis and the treatment or drug specified by the physician and places that information on a logically connected chart, such as a Web page. The connection may be based using one or more logic operators that may be indicative of various criteria for conducting analysis in a set of historical progress notes, such as chronological, pathological, pharmacological, and other criteria. Persons having authorized access to the system may be able to access the actual source document in the form of the progress note through the chart screen. As suggested above, the progress notes may be logically connected within a respective pathology and treatment therefor so that the physician or third party payor, e.g., health insurance carrier, can track a particular problem over its course in the life of a particular patient, or over any chosen chronological period. The progress notes can be transmitted into the system directly by the transcriber or transcription service thereby obviating the necessity of training and staffing the medical office for data input.

In one exemplary embodiment, data in the system is remotely accessible over the Internet using a respective computer using a web browser or by a cell phone with Internet access. Further, some phones with automated voice read back or handheld personal communication devices may be utilized to access information from the data storage device.

The medical records are organized or logically connected in a manner that facilitates easy usage by a remote caregiver such as an emergency room physician. The input of the medical data and the type of medical information utilized is selected to convey a snapshot of respective encounters between physician and patient to quickly alert the remote caregiver of any particular nuances in treatment of a patient, such as medical or drug sensitivity or the use of secondary drugs that may interact with primary drugs being taken by the patient. The system has appropriately restricted access and sufficient data encryption to prevent unauthorized persons from obtaining information from the medical database.

As shown in FIG. 1, system 10 may comprise a Web-based, distributed application that may be configured through a graphical user interface for uploading, or downloading, or both, one or more Web pages from a Web site 12, such as may be operated by the Assignee of the present invention. A server 14 comprises respective modules for managing business rules and data access to a database 16, such as a centralized database, operable to store patient and physician information. As suggested above, appropriate security measures, such as data encryption, and passwords are provided in server 14 to grant access to database 16 only to those users with pre-approved rights.

A plurality of participating or subscribing users, e.g., users 18, 20 and 22, will be able to communicate with server 14 via any suitable communications network, such as the Internet. Thus, users such as physicians, hospitals, pharmacies, ambulance services, emergency medical services, insurance companies, etc. will be able to query database 16 through server 14 from any device connected to the Internet that supports a suitable Web browser. In one advantage of the system of the present invention, due to its straightforward user interface, it is believed that users should be able to learn to navigate through the system after spending no longer than a few minutes. Users connecting to the web site will be asked to log on to gain access to the system. If they successfully log on, they will be able to query the system for information. Below are some examples of some of the users that will benefit from the system of the present invention. It will be appreciated, however, that the present invention is not limited to such users being that any

provider of medical services, or provider of services related to the health care industry will also benefit from the present invention.

Physicians

Physicians will primarily use the system to access patient records. They will be able to view past progress notes and validate newly transcribed progress notes. Physicians accessing the system will be able to search for patients, patient's past diagnoses and patient's past prescriptions. Physicians will also be able to view progress notes from patient visits to themselves as well as other physicians. Physicians will also be able to update their own information, their new patients information and update existing patient information, including patient insurance information.

Pharmacies

Pharmacies will be able to search for patients and view a patient's diagnosis and prescription history. They will also be able to update prescription information such as the date the prescription was filled. Pharmacies will also be able to view and update patient insurance information. Pharmacies will maintain their own billing/maintenance information.

Hospitals

Hospitals will generally use the system to search for patients. They will have the ability to view patient diagnosis and prescription history, as well as patient demographic information. Hospitals will also maintain their own billing/maintenance information.

Insurance Companies

By way of example, Insurance Companies may use the system to monitor progress notes of client patients. They will be able to search for physicians and view physician's diagnosis and prescription histories as well as the originating progress notes. Insurance Companies may use the system for multiple purposes, such as utilization review, quality assurance, grievance resolution, and including tracking of particular problems that may develop over the course of a long term condition. Insurance Companies will also maintain their billing/maintenance information.

Pharmaceutical Companies

Pharmaceutical Companies may use the system not necessarily to monitor prescription and diagnosis information of individuals but to have the capability of receiving composite reports analyzing what drugs are being collectively prescribed and how frequently. Thus, it is contemplated that Pharmaceutical Companies may not be granted access to any medical records of individuals. However, it is contemplated that Pharmaceutical Companies will benefit from gaining access to such composite reports. They will also maintain their billing/maintenance information.

Transcription Services

Transcription services will use the system to enter Physicians' progress notes. They will also be responsible for entering their maintenance information.

It will be appreciated that in order to provide maximum benefits, it is desirable for patient information to be accurate and up-to-date. As suggested above, the data will be centrally located and data storage may be executed at predefined time intervals, such as daily and even hourly. It will be further appreciated that as the size of the data managed by the system of the present invention grows, so will the need for broader bandwidth and greater system resources on the database servers. It will be recognized, however, that the techniques of present invention may be adaptable to advancing technologies and is not limited to presently available technology. Those of ordinary skill in the art will recognize that various uploading techniques may be used for updating the database with new patient information. However, regardless of the specific technique used, the updating should be accomplished in such a way as to minimize the time and effort by physicians, while getting the data into the system in a timely manner. Such uploading techniques may be used for entering other entity information, such as insurance companies, hospitals, pharmacies and pharmaceutical companies. It is envisioned that maintenance of this third party data will be done primarily by the entities themselves. Patient data, however, will be preferably maintained by physicians. Patient insurance information will also be preferably maintained by the patient's physician.

Exemplary Techniques for Updating Data

It is presently envisioned that in one exemplary embodiment there may be at least two different techniques for loading data from the transcription service into the database. One

technique will be for the transcribers to enter the transcription on the Web site. One or more web pages will be provided for transcribers to enter information directly into the system. This will allow the data to be immediately available for a physician's review. An alternate technique may be to provide the transcription services with a suitable word processing template, such as a MS Word or a Corel Word Perfect template. The template will have predefined fields for capturing progress note and patient data. Documents based on this template can be e-mailed to an e-mail address that is part of the Web site. A parsing module would be provided to parse the attachment and update the database with the progress note. As further advances in voice recognition occur, it is further envisioned that a physician could dictate using a device with voice-recognition capabilities and automatically generate an electronic file ready to be loaded into the database.

Ability to Enter and Search Progress Notes

The system is configured to allow respective transcribers to insert new progress notes from the transcription service. The physician who originally submitted the progress note for transcription will solely be allowed to review and validate the progress note. Once the physician has validated the progress note, that note is available for viewing only, to all users with the appropriate permissions. As further described below, these permissions are assigned to each user and affect each user's access to various system features.

Ability to Extract Prescriptions and Diagnoses from Progress Note

FIG. 2 illustrates a representation of a Web page 50 comprising an exemplary progress note 52. When the submitting physician reviews the progress note, he or she will identify on the progress note respective parameters selectable by that physician. One example of such parameters selectable by the physician may include a respective diagnoses from the progress note. The respective identified diagnosis may be added by the owner physician to a diagnosis list 54 by clicking on an icon 56 titled "Add Diagnosis". Another example of parameters selectable by the physician may include a respective prescription selected by the physician from the progress note. The respective identified prescriptions may then be added to a list of prescriptions 58 by clicking on an icon 60 titled "Add Diagnosis". If a diagnosis or prescription referenced in the progress note text also exists in the diagnosis or prescription list, the referenced item will be added to the diagnosis list or prescription list for that progress note. Each physician will have a respective list of diagnoses. The physician's progress note will generally comprise a

subset of this list. As shown in FIG. 2, progress note 52 also identifies the patient with an identifier, e.g., social security number or other suitable identifier, thus associating respective diagnoses with a respective patient. Any given physician will generally have one or more patients and respective patients of a given physician will have one or more progress notes and in turn each respective progress note will have one or more associated diagnoses. Similarly, each respective progress note will have zero or more associated prescriptions. Each of these diagnoses, or prescriptions, or both, can be related to the physician and the patient for searching. Further, each prescription record can be expanded to reference a pharmacy, a start date and an end date for the prescription. The system can thus determine at what pharmacy and on what date a prescription was filled for a given patient under a given physician's care, provided that all three users are enrolled in the system.

As shown in FIG. 3, database 16 may be populated with respective progress notes resulting from further encounters between the respective patient and any respective physician treating the patient so as to create a searchable historical set of progress notes for that respective patient. As suggested above, the set of historical notes is logically interconnectable based on one or more logic operators. As shown in FIG. 3, a list of diagnosis including a respective set of hyperlinks 72 each indicative of a respective diagnosis date is provided to enable a respective user to download and review the respective progress note associated with any given diagnosis date. Similarly, a set of hyperlinks 74 may be listed in a desired chronological order to enable the user to monitor progress of a given pathological condition in the patient. FIG. 3 further shows a list of prescriptions including a respective set of hyperlinks 76 each indicative of a respective prescription date that allows the user for logically connecting respective progress notes and prescription dates.

Ability to Search Progress Notes by Diagnosis

As suggested above, physician and users, such as insurance companies, will have the ability to search for progress notes that discussed a specific diagnosis. The user will be presented with a list of diagnoses for a given patient. That list will name the diagnosis, the date of the first occurrence and the date of the most recent occurrence. Selecting one of these items will take the user to a read-only display of the progress note. Here, the user will have the ability to scroll through the progress notes pertaining to this diagnosis.

Ability to Search Progress Notes by Prescription

Similarly, such users will have the ability to search for progress notes that mention a specific prescription. The user will be presented with a list of prescriptions for a given patient. That list will name the diagnosis, the date of the first occurrence and the date of the most recent occurrence. Selecting one of these items will take the user to a read-only display of the progress note. Here, the user will have the ability to scroll through the progress notes pertaining to this prescription.

Ability to Search for Patients

Users will have the ability to search for a patient by the patient's social security number, date of birth, last name and first name. Any combination of these criteria may be used to search for a patient. If only the date of birth field is populated, the search results will contain all patients with that date of birth. If date of birth and last name are populated, then the search results will contain all patients with the date of birth the user entered, having the last name the user entered. If no patients meet the search criteria, a message will display to the user to let them know that no such patient exists and they may want to broaden their search requirements. If multiple patients meet the search criteria, a table will display with each row in the table corresponding to a different patient. The columns will be First Name, Last Name, Social Security Number and Date of Birth.

An example of a table comprising searchable parameters in connection with patient identification is listed in Table 1 below:

First Name	Last Name	Social Security Number	Date of Birth
John	Doe	123121234	01/01/1950
Juan	Don	OC0000017	03/15/1947

Table 1

It will be appreciated that patients who are not citizens of the United States may not have an assigned social security number. Thus, a module may be provided for tracking patients who are not United States citizens. For example, an alphabetical character may be inserted into the SSN as an indicator that the patient is not a citizen of the US and in that case the search may be conducted by name and date of birth.

Ability to Search Physicians and other Users

Users will have the ability to search for physicians either by name or location, or both. Searching by name will return all physicians in the system where the text entered in the search matches text the physician's name. Searching by location may be broken down by city and postal code. Either field may be used. For example, if both fields contain text, then physicians in the system having an address where the city matches text entered in the city field and a postal code matching text entered in the postal code field, will be returned. Users will further have the ability as described above to search for other users, such as pharmacies, hospitals, insurance companies, transcription services either by name or location, or both.

Ability to Add and Update Patient Demographic Information

In one exemplary embodiment of the system of the present invention, physicians and hospitals will have the ability to modify patient demographic data. The purpose of this functionality is to allow physician's offices and hospitals to have the latest patient demographic information. This information is to include the patient's name, social security number, date of birth, and any other useful contact information. By way of example, contact information may include the patient's address or addresses and their phone number(s) and email addresses, if available. In addition to the standard contact information, the patient's insurance information may also be tracked. Since a given patient may have multiple insurance companies, the system can support the addition and removal of multiple insurance companies for patients. There may also be provided additional fields to enter new insurance companies and make any appropriate changes to insurance company information for those that are already in the system.

Ability to Add and Update Hospital Information and other Users

In one exemplary embodiment of the system of the present invention, physicians or hospitals users can add hospitals into the system. When a physician adds a hospital to the system they do not necessarily create a user for that hospital. Physicians and pharmacies can enter hospital billing/maintenance data, as well as the hospitals. Further, insurance companies can be similarly added by physicians when creating new patient or updating existing patient's information. Similarly, pharmacies can be added by physicians when creating new patient or updating existing patient's prescription.

Validate progress note

When a physician logs into the system, they will be presented a list of hyperlinks to all progress notes containing their respective UPIN identifier that have not been validated. When the physician clicks a respective hyperlink, they will be taken to a Web page displaying an editable version of the progress note. The physician will have the ability to update any information contained within the progress note. In addition, the physician will be able to select text from within the progress note and add this text to the diagnosis list or the prescription list. These lists will be populated with the existing diagnoses and prescriptions for the physician. If text within the progress note text matches text in the diagnosis or prescription lists, that text will become selected in the list. This is an indication that the diagnosis or prescription is associated with the progress note. Once the physician submits this progress note, it is available to any other entities with proper permission, to view.

Entering/Updating Billing/Maintenance information

Users will have access to a page where they can enter their address and contact information. This page will allow users with proper permission to edit their information and other users to view it.

Security and Record Access

It should be apparent that security is important in this type of system due to the sensitivity of patient data. The security for this system will comprise multiple levels. The system will use SSL (Secure Socket Layer) 128 bit encryption. This will ensure that data going between the clients and the web server is encrypted. It is believed that such encryption should prevent hackers from intercepting packets and getting sensitive information. Each Physician, Pharmacy, hospital and pharmaceutical company in the system, will have a user account to gain access. By way of example, the account may comprise a user name, password and entity type; i.e. Physician, hospital, Pharmacy, etc. The user name may require a minimum number of characters, e.g., at least six characters long and may not include the <space> character. Similarly, the password may be at least six characters long and may be case sensitive. The password may be chosen to contain at least one lower and one upper case letter and at least one numeric character. A log of system access attempts may be maintained. This log will consist of user name, password, IP address and date time stamp. This log can be used to determine if there have been unauthorized

attempts and if so, where they originated. Tables 2-7 below illustrate exemplary access rights for typical entities that may access the system. An X in the field indicates that the entity has privileges for that task.

Physician:

	Search	View	Insert	Update
Diagnosis	X	X	X	X
Prescription	X	X	X	X
Progress Note	X	X	X	X
Entity Maintenance				
Patient Demographics	X	X	X	X
Physician	X	X	X	X
Pharmacy	X	X		
Hospital	X	X		
Insurance Company	X	X	X	X
Pharmaceutical Company	x			
Transcription Service	X	X	X	X

Table 2

Pharmacy:

	Search	View	Insert	Update
Diagnosis	X	X		
Prescription	X	X		
Progress Note				
Entity Maintenance				
Patient Demographics		X		X
Physician		X		
Pharmacy	X	X	X	X
Hospital				
Insurance Company		X		
Pharmaceutical Company				
Transcription Service				

Table 3

Hospital:

	Search	View	Insert	Update
Diagnosis	X	X		
Prescription	X	X		
Progress Note				
Entity Maintenance				
Patient Demographics	X	X		X
Physician		X		
Pharmacy				
Hospital	X	X	X	X
Insurance Company		X	X	X
Pharmaceutical Company				
Transcription Service				

Table 4

Insurance Company:

	Search	View	Insert	Update
Diagnosis	X	X		
Prescription	X	X		
Progress Note	X	X		
Entity Maintenance				
Patient Demographics	X	X		
Physician	X	X		
Pharmacy	X	X		
Hospital	X	X		
Insurance Company	X	X	X	X
Pharmaceutical Company				
Transcription Service				

Table 5

Pharmaceutical Company:

	Search	View	Insert	Update
Diagnosis				
Prescription				
Progress Note				
Entity Maintenance				
Patient Demographics				
Physician				
Pharmacy				
Hospital				
Insurance Company				
Pharmaceutical Company			X	X
Transcription Service				

Table 6

Transcription Service:

	Search	View	Insert	Update
Diagnosis				
Prescription				
Progress Note				
Entity Maintenance				
Patient Demographics				
Physician				
Pharmacy				
Hospital				
Insurance Company				
Pharmaceutical Company				
Transcription Service			X	X

Table 7

Tracking/Billing Module

The system of the present invention includes a module for tracking system usage as a function of entity. As patients are added into the system, each patient will include a primary physician who is a participant in the services provided by the assignee of the present invention. When a user or entity that is not a patient's primary physician accesses that patient's records, that entity may be charged a predefined access fee. The participant who is the patient's primary physician may receive a predefined percentage of the access fee. If the participant was referred to the system by another physician, that referrer may also receive a predefined percentage of the access fee. When an attempt is made to access a record, the tracking module will first determine whether the entity has permission to view the record. If the entity does not have permission, the entity is then compared to the record's owner. If they are the same, no accounting transaction is registered. If they are different, however, the entity accessing the record is a charged a predefined fee in accordance with a predefined entity type billing schedule stored in memory. The owner of the record would then receive a payment record of a predefined percentage of the access charge. The tracking module would then check for respective parents of the owner. If the parent has an owner, that owner level is checked, and they would receive a payment record based on a predefined billing schedule. Then the tracking module would check for parents of the parent and continue with any further iterations until there is no parent record for the present parent.

FIG. 4 illustrates an exemplary layout of a data structure 100 of a plurality of computer-readable records or pages including respective data fields for securely and accurately managing medical records of a plurality of patients. For example, respective data fields of a computer-readable page comprising data indicative of a respective physician or doctor would be Doctor_Id, First Name, Last Name, UPIN, FEI, Parent_Id and Pay_Level_ID. Similarly, respective data fields of a computer-readable page comprising a progress note would be Note_Id, Patient_Id, Note_Date/Time, Doctor_Id, Note, Valid, Transcript_Id.

As shown in FIG. 4, the computer-readable progress note page may be logically interconnected to a computer-readable progress problem page and/or a progress description page. The progress problem page may be in turn connected to a computer-readable page comprising a problem list including a list of inactive problems. Similarly, the computer-readable progress prescription page may be logically interconnected to a computer-readable page comprising a prescription list including a list of inactive prescriptions. Each of the users has a respective computer-readable address page comprising data indicative of address information for each user. A computer-readable page comprising data indicative of patient information may be linked to a computer-readable page comprising patient insurance information and in turn to a page comprising insurance company data of a given patient. The type of user may be identified based on entity type. In addition, each user may have a corresponding computer-readable page comprising contact information. As suggested above, the tracking module may track system access and the type of entity gaining such access to generate billing information including payable and receivable accounts and including type of payment, such as check payment, credit card payment, etc. A computer-readable applicant page may comprise respective data fields corresponding to a prospective user of the system.

The present invention can be embodied in the form of computer-implemented processes and apparatus for practicing those processes. The present invention can also be embodied in the form of computer program code containing computer-readable instructions embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other computer-readable storage medium, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. The present invention can also be embodied in the form of computer program code, for example, whether stored in a storage medium, loaded into and/or executed by a computer, or transmitted over some transmission

medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. When implemented on a general-purpose computer, the computer program code segments configure the computer to create specific logic circuits or processing modules.

Appendix I comprises exemplary COM interfaces including flow diagrams which may be utilized to control operation of a system embodying the present method for managing medical records. By way of example and not of limitation, the interfaces in Appendix I were written using MS Visual Basic syntax.

It will be understood that the specific embodiment of the invention shown and described herein is exemplary only. Numerous variations, changes, substitutions and equivalents will now occur to those skilled in the art without departing from the spirit and scope of the present invention. Accordingly, it is intended that all subject matter described herein and shown in the accompanying drawings be regarded as illustrative only and not in a limiting sense and that the scope of the invention be solely determined by the appended claims.

WHAT IS CLAIMED IS:

1. A medical health record storage and retrieval system comprising:

an interface module operable to extract a patient's medical diagnosis and treatment from respective progress notes of a physician;

a storage module configured to store the extracted diagnosis and treatment in a logically connected database;

a server configured to allow access to the stored database by authorized users; and

a processor module configured to track users accessing the database and to bill the accessing users for each access of the database.

2. A computerized method for managing respective health records of a plurality of patients, said method comprising:

uploading a progress note of a respective patient, said progress note comprising data relative to an encounter between a respective physician and the respective patient;

identifying on said progress note respective parameters selectable by the respective physician;

storing said progress note with said identified parameters in a database accessible to a plurality of authorized users; and

populating said database with respective progress notes resulting from further encounters between the respective patient and any respective physician so as to create a historical set of progress notes for that respective patient.

3. The computerized method of claim 2 wherein the identified parameters are selected to convey a snapshot of said encounter.

4. The computerized method of claim 2 wherein the identified parameters are selected from the group of consisting of diagnosis and prescription parameters.

5. The computerized method of claim 2 wherein the set of historical progress notes is interconnectable based on one or more logic operators.

6. The computerized method of claim 5 wherein one of the logical operators comprises a chronology- indicative operator.

7. The computerized method of claim 5 wherein one of the logical operators comprises a pathology-indicative operator.

8. The computerized method of claim 5 wherein one of the logical operators comprises a pharmacology-indicative operator.

9. The computerized method of claim 1 further comprising tracking users accessing the database to process respective billing of the accessing users for each access of the database.

10. The computerized method of claim 2 wherein the database is accessible to the plurality of users through a communications network.

11. The computerized method of claim 10 wherein the communications network comprises the Internet.

12. A computer-readable medium encoded with computer program code for managing respective health records of a plurality of patients, the program code causing a computer to execute a method comprising:

uploading a progress note of a respective patient, said progress note comprising data relative to an encounter between a respective physician and the respective patient;

identifying on said progress note respective parameters selectable by the respective physician;

storing said progress note with said identified parameters in a database accessible to a plurality of authorized users; and

populating said database with respective progress notes resulting from further encounters between the respective patient and any respective physician so as to create a historical set of progress notes for that respective patient, the set of historical progress notes being interconnectable based on one or more logic operators.

13. The computer-readable medium of claim 12 wherein the identified parameters are selected to convey a snapshot of said encounter.

14. The computer-readable medium of claim 12 wherein the identified parameters are selected from the group of consisting of diagnosis and prescription parameters.

15. The computer-readable medium of claim 12 wherein one of the logical operators comprises a chronology- indicative operator.

16. The computer-readable medium of claim 12 wherein one of the logical operators comprises a pathology-indicative operator.

17. The computer-readable medium of claim 12 wherein one of the logical operators comprises a pharmacology-indicative operator.

18. The computer-readable medium of claim 12 further comprising tracking users accessing the database to process respective billing of the accessing users for each access of the database.

19. The computer-readable medium of claim 12 wherein the database is accessible to the plurality of users through a communications network.

20. The computer-readable medium of claim 19 wherein the communications network comprises the Internet.

21. A medical health record storage and retrieval system comprising:

means for extracting a patient's medical diagnosis and treatment from respective progress notes dictated by a physician;

means for storing the extracted diagnosis and treatment in a logically connected database;

means for allowing access to the stored database by authorized users; and

means for tracking users accessing the database and for billing the accessing users for each access of the database.

COMPUTERIZED METHOD AND SYSTEM
FOR OBTAINING, STORING
AND ACCESSING MEDICAL RECORDS

ABSTRACT OF THE DISCLOSURE

Computerized method and system for managing respective health records of a plurality of patients are provided. The method allows for uploading a progress note of a respective patient. The progress note includes data relative to an encounter between a respective physician and the respective patient. The method further allows for identifying on the progress note respective parameters selectable by the respective physician. A storing step allows for storing the progress note with the identified parameters in a database accessible to a plurality of authorized users. A populating step allows for populating the database with respective progress notes resulting from further encounters between the respective patient and any respective physician so as to create a historical set of progress notes for that respective patient.

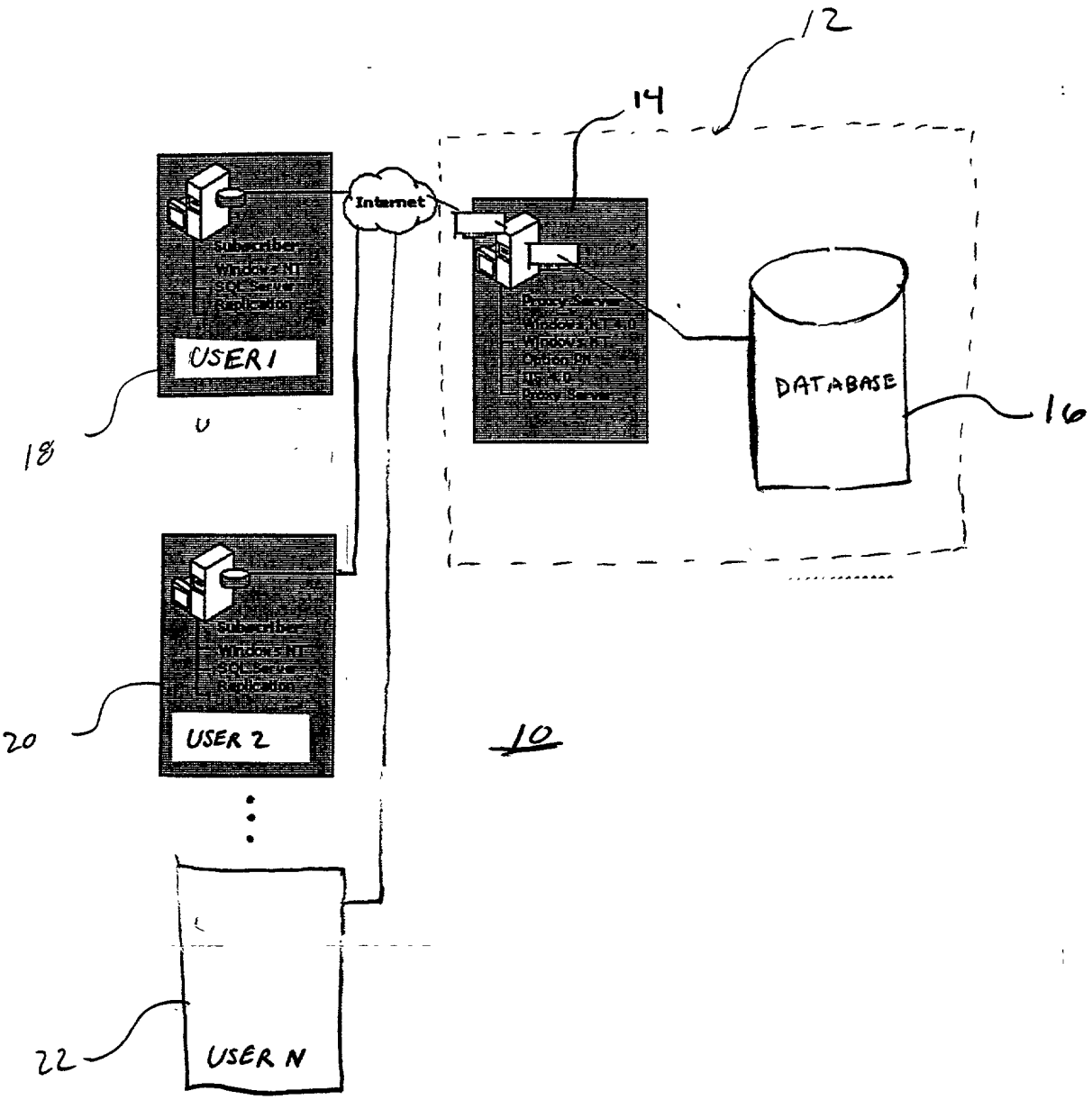


FIG. 1

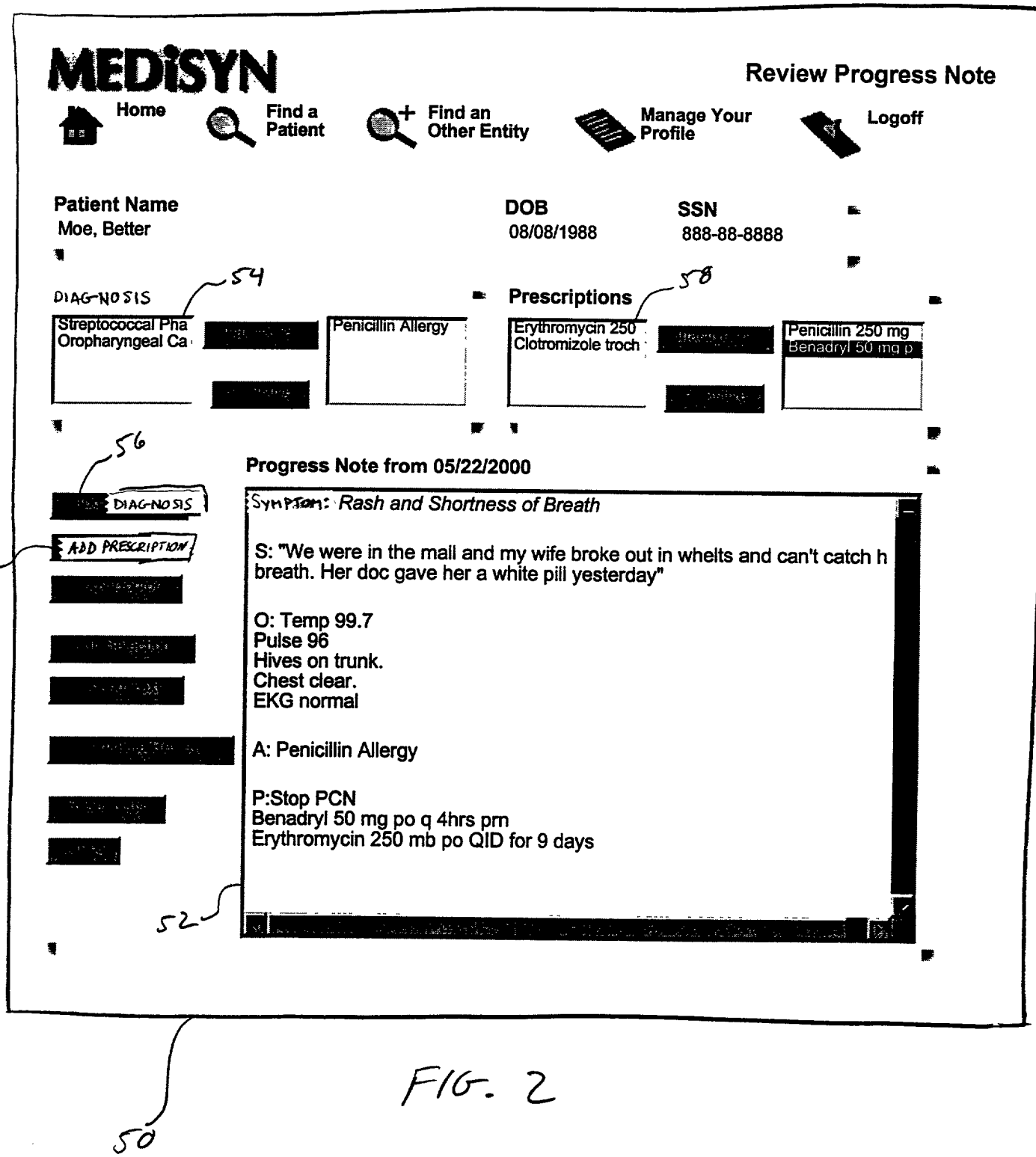

MEDISYN
Review Progress Note
Home
Find a Patient
Find an Other Entity
Manage Your Profile
Logoff
Patient Name
Moe, Better
DOB
08/08/1988
SSN
888-88-8888
DIAGNOSIS
54
Streptococcal Pha
Oropharyngeal Ca
Penicillin Allergy
Prescriptions
58
Erythromycin 250
Clotromizole troch
Penicillin 250 mg
Benadryl 50 mg p
56
DIAGNOSIS
ADD PRESCRIPTION
60
Progress Note from 05/22/2000
Symptom: Rash and Shortness of Breath
S: "We were in the mall and my wife broke out in whelts and can't catch h
breath. Her doc gave her a white pill yesterday"
O: Temp 99.7
Pulse 96
Hives on trunk.
Chest clear.
EKG normal
A: Penicillin Allergy
P:Stop PCN
Benadryl 50 mg po q 4hrs prn
Erythromycin 250 mb po QID for 9 days
52
50

FIG. 2

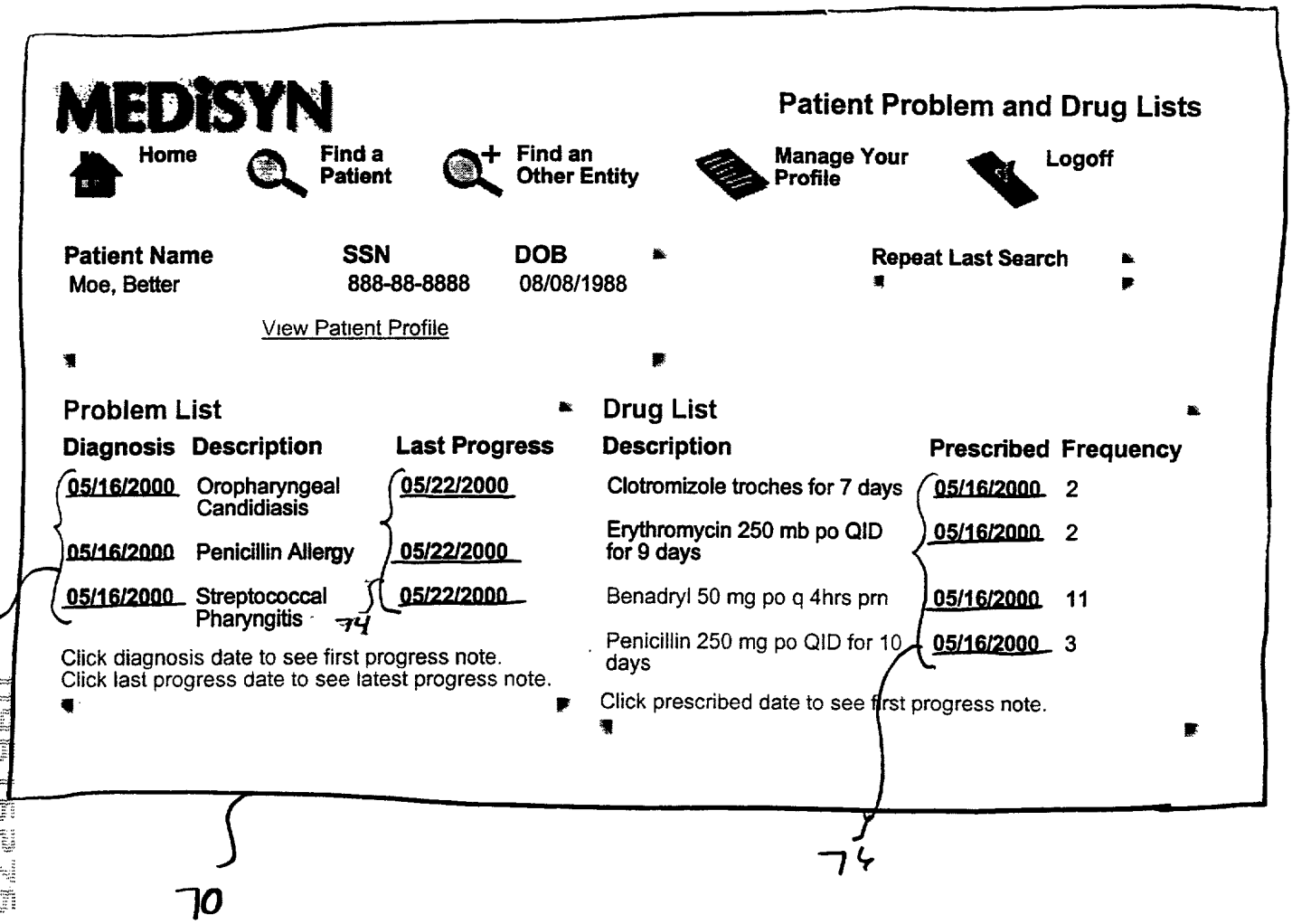

MEDISYN
Patient Problem and Drug Lists
Home
Find a Patient
Find an Other Entity
Manage Your Profile
Logoff
Patient Name
SSN
DOB
Repeat Last Search
Moe, Better
888-88-8888
08/08/1988
View Patient Profile
Problem List
Diagnosis
Description
Last Progress
05/16/2000
Oropharyngeal Candidiasis
05/22/2000
05/16/2000
Penicillin Allergy
05/22/2000
05/16/2000
Streptococcal Pharyngitis
05/22/2000
Click diagnosis date to see first progress note.
Click last progress date to see latest progress note.
Drug List
Description
Prescribed
Frequency
Clotromizole troches for 7 days
05/16/2000
2
Erythromycin 250 mb po QID for 9 days
05/16/2000
2
Benadryl 50 mg po q 4hrs prn
05/16/2000
11
Penicillin 250 mg po QID for 10 days
05/16/2000
3
Click prescribed date to see first progress note.
12
74
74
70

FIG. 3

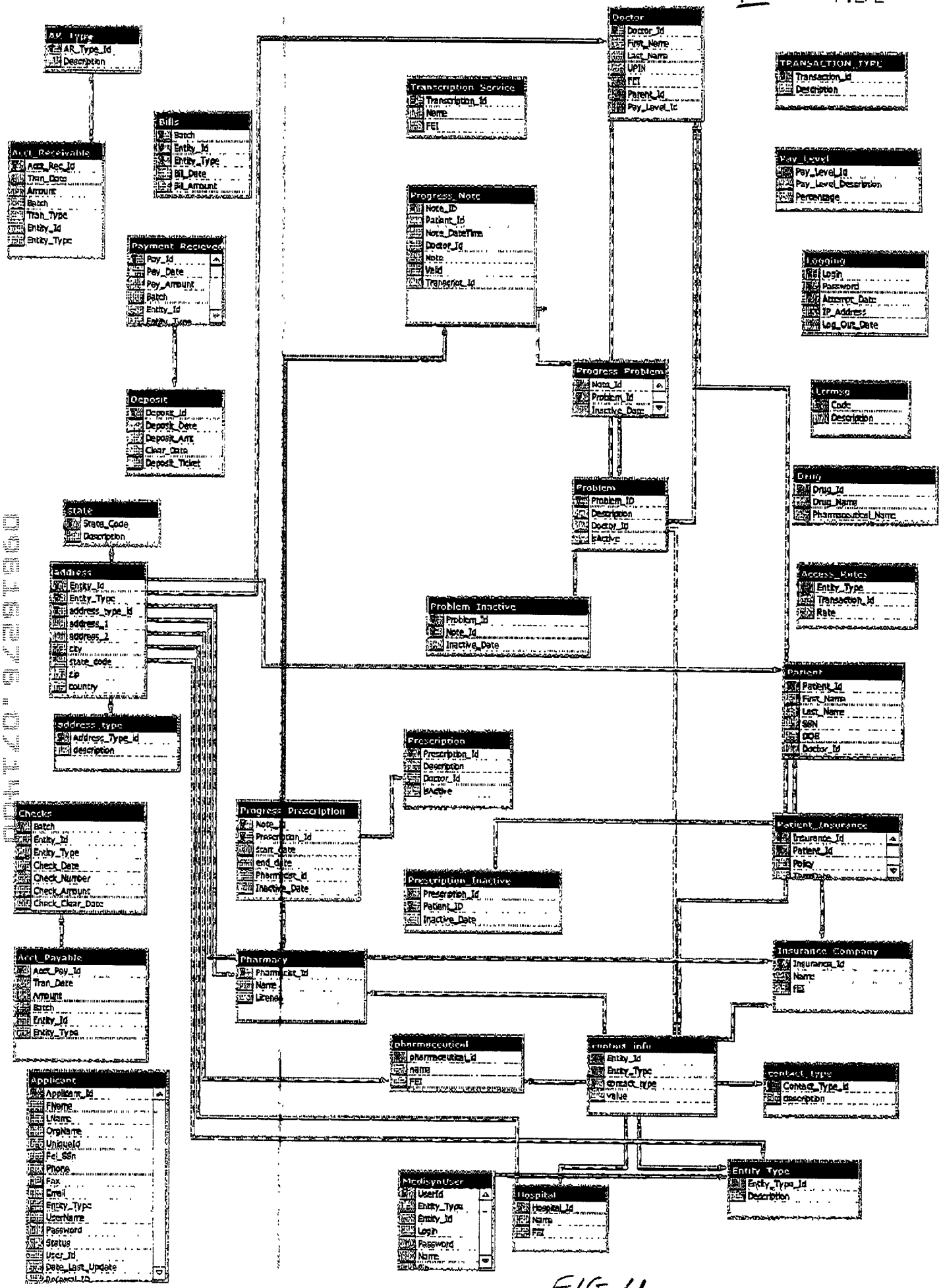

AR_Type
AR_Type_Id
Description
Acct_Receivable
Acct_Rec_Id
Tran_Date
Amount
Batch
Tran_Type
Entity_Id
Entity_Type
Bills
Batch
Entity_Id
Entity_Type
Bill_Date
Bill_Amount
Payment_Recieved
Pay_Id
Pay_Date
Pay_Amount
Batch
Entity_Id
Entity_Type
Deposit
Deposit_Id
Deposit_Date
Deposit_Amt
Clear_Date
Deposit_Ticket
State
State_Code
Description
Address
Entity_Id
Entity_Type
address_type_id
address_1
address_2
city
state_code
zip
country
address_type
Address_Type_Id
description
Checks
Batch
Entity_Id
Entity_Type
Check_Date
Check_Number
Check_Amount
Check_Clear_Date
Acct_Payable
Acct_Pay_Id
Tran_Date
Amount
Batch
Entity_Id
Entity_Type
Applicant
Applicant_Id
FName
LName
OrgName
UniqueId
Fed_SSn
Phone
Fax
Email
Entity_Type
UserName
Password
Status
User_Id
Date_Last_Update
Doctor
Doctor_Id
First_Name
Last_Name
UPIN
FEI
Parent_Id
Pay_Level_Id
Transcription_Service
Transcription_Id
Name
FEI
Progress_Note
Note_ID
Patient_Id
Note_DateTime
Doctor_Id
Note
Valid
Transcript_Id
Progress_Problem
Note_Id
Problem_Id
Inactive_Date
Problem
Problem_ID
Description
Doctor_Id
Inactive
Problem_Inactive
Problem_Id
Note_Id
Inactive_Date
Prescription
Prescription_Id
Description
Doctor_Id
Inactive
Progress_Prescription
Note_Id
Prescription_Id
start_date
end_date
Pharmacist_Id
Inactive_Date
Prescription_Inactive
Preseription_Id
Patient_ID
Inactive_Date
Pharmacy
Pharmacist_Id
Name
License
pharmaceutical
pharmaceutical_Id
name
FEI
Contact_Info
Entity_Id
Entity_Type
Contact_Type
value
MedisynUser
UserId
Entity_Type
Entity_Id
Login
Password
Name
Hospital
Hospital_Id
Name
FEI
TRANSACTION_TYPE
Transaction_Id
Description
Pay_Level
Pay_Level_Id
Pay_Level_Description
Percentage
Logging
Login
Password
Attempt_Date
IP_Address
Log_Out_Date
Lrcmsg
Code
Description
Drug
Drug_Id
Drug_Name
Pharmaceutical_Name
Access_Rates
Entity_Type
Transaction_Id
Rate
Patient
Patient_Id
First_Name
Last_Name
SSN
DOB
Doctor_Id
Patient_Insurance
Insurance_Id
Patient_Id
Policy
Insurance_Company
Insurance_Id
Name
FEI
Contact_Type
Contact_Type_Id
description
Entity_Type
Entity_Type_Id
Description

FIG. 4

APPENDIX I

MediSyn COM Interfaces

MediSyn COM Interfaces

Overview:

The following is a list of the interfaces the COM component will support. Each of the following interface definitions is composed of the name, followed by the method description, the signature, the parameters and the return type. The table of contents lists each interface followed by their methods. Each method will have a brief description. The names of the methods within the table of contents are hyperlinks to the method description detail.

The description will describe the functionality of the method. This will include any constraints or preprocessing information necessary to call this method.

The signature, for the purposes of this document, is written using MS Visual Basic syntax. This facilitates readability for the developers on this project, as the middle tier component will be developed using MS Visual Basic.

The parameter's names each parameter in the method call and gives a brief description of the purpose of the parameter. If the parameter has specific domain constraints, they will be noted in an indented small font beneath the parameter description.

The return type will be consistent for each method. The return data type will be named, followed by the rules of what data will be returned.

***note: All ADODB. Recordsets may be handled as 2 dimensional arrays, rather than a recordsest.*

Table of Contents

getCodeTable - Returns the specified code table. Used to populate dropdown lists.

IProblem_Drugs

getDrues - Returns list of drugs.
updateDrugs - Updates prescriptions, including start date and end date and active flag.
getProblems - Returns list of problems.
updateProblems - Updates problems, including active flag.

IEntity

getEntity - Returns the specified entity's record.
updateEntity - Updates the specified entity's record.
getPatientInsurance - Returns patient insurance records.
updatePatientInsurance- Updates the patient's insurance information.

IContact

getAddressInfo - Returns a list of address records for an entity.
updateAddressInfo - Updates an entity's address record.
getContactInfo - returns a list of contact records for an entity.
updateContactInfo - Updates an entity's contact info record.

IAccounting

bill - Creates an AR records.
getBills - Generates bills from AR records and returns a list of bill records.
getAcctsPavable - Returns a list of AP Records.
proceessChecks - Creates check records based on AP records.
processReceipts - Creates receipt records which are applied to existing bill records.
getReceipts - Returns a list of receipt records.
updateDeposit - Creates a deposit record based on existing receipt records.
getReconciliation - Returns deposit and check records that have not cleared.
updateReconciliation- Updates deposit and check records that have cleared.

COM Methods

getApplicants

Description: This method will return the specified applicant or all applicants in the applicant's table. The number of records returned will be limited to 20. This will ensure that the web interface is not overwhelmed with hundreds of records.

Signature:

```
Public Function getApplicants(rsApplicants as ADODB.Recordset,
                              Optional iApplicantId as Integer) as Long
```

Parameters:
rsApplicants: Contains applicant records.
iApplicantId: specifies the applicant to be returned.

The following columns are returned: Applicant Id, Entity Type, FName, LName, OrgName, UniqueId, FEI_SSN, Phone, Fax, Email, Entity Type, UserName, Password.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateApplicants

Description:
This method will create a MediSyn user from an applicant and flag that applicant as accepted, or flag the applicant as denied if rejected. Once an applicant has been passed to updateApplicant their record in the applicant table is flagged as approved or denied and the date of the transaction is recorded. 60 days after this date, the record will be removed. The record will stay in the database for 60 days for auditing purposes. After that time, any issues with the applicant should have been resolved

Signature:
Public Function updateApplicants(iApplicantId() as Integer, _
Optional isApproved() as Boolean) as Long

Parameters:
iApplicantId(): Array, specifies the applicant to be updated. The isApproved flag will determine whether the applicant becomes a member of MediSyn.
isApprovedo: If true a new MediSyn user is created and the applicant record is flagged as approved. If false the applicant record is flagged as denied.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

newApplicant

Description:
This method will create a new applicant and if the isApproved optional parameter = true, a new user. All of the parameters with valid data will be used to populate the applicant table and if the applicant is approved, the MediSynUser table, the corresponding entity table as specified by iEntityType, and the Contact_Info table.

Signature:
Public Function newApplicant(sFName as String, _
sLName as String, _
sOrgName as String,
sUniqueId as String,
sFEI-SSN as String, -

sPhone as String, _
sFax as String, _
sEMail as String, _
iEntityType as Integer, _
sUserName as String, _
sPassword as String, _
Optional iReferrer as Integer, _
Optional isApproved as Boolean) as Long

Parameters:
sFName : Specifies the first name if the applicant is a doctor.
sLName : Specifies the last name if the applicant is a doctor.
sOrgName : Specifies the organization name if the applicant is not a doctor.
sUniqueId : Specifies the unique identifier for the entity being added. UPIN for doctor, license forPharmacy.
sFEI SSN : Specifies the SSN or FEI for the entity being added.
sPhone : Specifies the phone number for the entity being added.
sFax : Specifies the fax number for the entity being added.
sEMail : Specifies the email address for the entity being added.
iEntityType : Specifies the entity type for the entity being added.
sUserName : Specifies the user name for the entity being added.
sPassword : Specifies the password for the entity being added.
iReferrer : Specifies the doctor _id of the doctor who will be the parent id for the doctor being added.
isApproved: If true a new MediSyn user is created and the applicant record is deleted. If false the applicant record is deleted.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

login

Description:
The login method provides a mechanism for MediSyn users to access the application.

Signature:
Public Function login(sLogin as String, _
sPassword as String, _
sIP_Address as String, _
ByRef iEntityType as Integer, _
ByRef iEntityId as Integer) as Long

Parameters:
sLogin: Specifies the username of the applicant. This value cannot contain <space> characters and must be a minimum of 6 and a maximum of 15 characters in length.
sPassword: Specifies the password of the applicant. This value cannot contain <space> characters and must be a minimum of 6 and a maximum of 15 characters in length. It

must also contain at least one uppercase and one lowercase letter and at least one numeric character.

sIP_Address: Specifies the IP Address of the user attempting to login.
iEntityType: The type of entity of the user logging in. This is returned to the caller.
iEntityId: The id of entity of the user logging in. This is returned to the caller.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

logout

Description:
The logout method provides a mechanism to inform MediSyn that a user has left the application. This is used for logging purposes. By noting the date, time and location of the login attempt, the application can track when and what lengths of time users have used the system.

Signature:
```
Public Function logout(ByVal iEntityId as Integer, _
                       ByVal iEntityType as Integer ) as Long
```

Parameters:
iEntityId: Specifies the id of the entity logging out.
iEntityType: Specifies the type of user logging out.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getUser

Description:
This method will return the specified user from the MediSynUser table. Administrators will use this method when a user forgets their login or password and they do not have an email address.

Signature:
```
Public Function getUser(ByVal iEntityType as Integer, _
                        sUniqueId as String, _
                        sLogin as String, _
                        sPassword as String ) as Long
```

Parameters:
iEntityType: Specifies the entity whose info is to be sent
sUniqueId: Specifies the unique identifier for the type of user defined by iEntityType above.
Examples: Doctor - UPIN, Pharmacy - License, FEI - Hospital.
sLogin - Where the user's login name is returned.
sPassword - Where the user's login name is returned.

Return type:

Long: returns 0 if successful and an Error Code if unsuccessful.

updateUser

Description:
This method updates user info to allow the user to change their login name and/or password.

Signature:
```
Public Function updateUser(ByVal iEntityId as Integer, _
                           ByVal iEntityType as Integer, _
                           Optional sUserName as String, _
                           Optional sPassword as String ) as Long
```

Parameters:
iEntityId: Specifies the entity whose info is to be updated.
iEntityType: Specifies the entity whose info is to be updated
sUserName: Specifies the new username
sPassword: Specifies the new password.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

sendUserInfo

Description:
This method gets user info and sends the user their login name and password via email. This provides the same functionality as getUser, except that it does not require the administrator's involvement.

Signature:
```
Public Function sendUserInfo(ByVal iEntityType as Integer, _
                             sUniqueId as String ) as Long
```

Parameters:
iEntityType: Specifies the entity whose info is to be sent
sUniqueId: Specifies the unique identifier for the type of user defined by iEntityType above.
Examples: Doctor - UPIN, Pharmacy - License, FEI - Hospital.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getNotes

Description:
This method allows the interface to receive all of the progress notes for a physician, patient, drug, or problem. The collection of notes returned is determined by the iNoteType parameter. This value will resolve to referred notes, open notes, problem notes, or drug notes.

Signature:

```
Public Function getNotes(ByVal iEntityId as Integer, _
                         iNoteType as Integer,
                         rsEntityNotes as ADODB_Recordset, _
                         Optional iDrug_Problem as Integer ) as Long
```

Parameters:

iEntityId: Specifies the doctor who is logged into the system. If iNoteType = 1 or 2, the iEntityId points to doctor _id. If iNoteType = 3 or 4, the iEnftyId points to patient id

iNoteType: Specifies the progress notes to return.

1 -referred notes
2 - open notes
3 - drug notes
4 - problem notes

rsReferringNotes: An ADO Recordset which will contain all of the progress notes where iDoctorId is the referring doctor id.

iDrug Problem: This field will contain the drug id or problem id if the iNoteType is 3 or 4 respectively.

The following columns are returned: note_id, patient.first_name, patient.last_name, note_date, note_time, ctor.first_name, doctor.last_name

Return type:

Long: returns 0 if successful and an Error Code if unsuccessful.

<u>getProgressNote</u>

Description:

The two dimensional array or ADODB.Recordset, rsProgressNote, returned will contain the fields in the progress note identified by iNoteId. The two dimensional array or ADODB.Recordset, rsDrugs, returned will contain the drug records associated with the progress notes identified. The two dimensional array or ADODB.Recordset, rsProblem, returned will contain the problems associated with the progress note identified.

Signature:

```
Public Function getProgressNote(ByVal iNoteId as Integer, _
                                rsProgressNote as ADODB.Recordset, _
                                rsDrugs as ADODB.Recordset, _
                                rsProblems as ADODB.Recordset ) as Long
```

Parameters:

iNoteId: Specifies the patient for whom the demographic information is requested.

rsProgressNote: An ADO Recordset which will contain the referenced progress note.

The following columns are returned: note_id, note_date, note_time, doctor_id, note, referring_Doctor, valid, transcript_id.

rsDrugs: An ADO Recordset which will contain those drugs referenced in the progress note.

The following columns are returned: drug_id, description, isActive.

rsProblems: An ADO Recordset which will contain those problems referenced in the the referenced progress note.

The following columns are returned: problem_id, description, isActive.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateProgressNote

Description:
Adds or updates a progress note for an existing patient and doctor. If the iNoteId parameter is 0, then the progress note is coming from the transcription service and is to be inserted into the Progress _Note table. If the iNoteId parameter > 0, then the note is being validated by the doctor and will be updated with the problems and drugs associated with the note. Also, a progress note may only be updated once. At that time the valid flag is set to true and the record can no longer be modified.

Signature:
```
Public Function updateProgressNote (ByVal iPatientId as Integer,
                                    ByVal iNoteId as Integer,
                                    ByVal iDoctorId as Integer,
                                    ByVal iTranscriptionId as Integer,
                                    sNoteDate as String,
                                    sNoteText as String,
                                    Optional aProblems() as Variant,
                                    Optional aDrugs() as Variant,
                                    Optional sNoteTime as String,
                                    Optional ByVal iRefDoctorId as Integer) as Long
```

Parameters:
iPatientId: Specifies the patient referenced in the progress note.
iNoteId: Specifies the note id of the progress note. If this value is zero, then a new progress note is created.
iDoctorId: Specifies the doctor referenced in the progress note.
iTranscriptionId: Specifies the transcription service referenced in the progress note.
sNoteDate: Specifies the date of the progress note.
sNoteText: Specifies the text of the progress note.
aProblems: Two-dimensional array specifying the problems in the note. The columns are problemId, description, active. If the problemId is 0, then the problem is inserted into the problem table. All problems should be tagged to the progress note for future reference.

aDrugs: Two-dimensional array specifying the Drugs in the note. The columns are DrugId, description, active. If the DrugId is 0, then the drug is inserted into the drugs table. This array does not have to contain data. There may or may not be a prescription associated with a progress note. All drugs should be tagged to the progress note for future reference.
sNoteTime: Specifies the time of the progress note.
iRefDoctorId: Specifies the referring physician for the progress note.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

findPatient

Description:
The two dimensional array or ADODB.Recordset returned will contain the patient's name, SSN and DOB. This will be presented to. the user as a hyperlink to view the patient's problem(diagnosis) and drug(prescription) lists. At least one of the fields is required to have a value.

Signature:
Public Function findPatient(rsPatients as ADODB. Recordset,
Optional sFName as String, _
Optional SLName as String, _
Optional sSSN as String, _
Optional sDOB as String) as Long

Parameters:
rsPatients: An ADO Recordset which will contain all of the patient records that matched the search criteria.
sFName: Specifies the first name of the patient being searched.
sLName: Specifies the last name of the patient being searched.
sSSN: Specifies the SSN of the patient being searched.
sDOB: Specifies the DOB of the patient being searched.

The following columns are returned: patient_id, patient.first_name, patient.last_name, patient_ssn,, patient_dob, doctor_id.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

search

Description:
The two-dimensional array or ADODB.Recordset returned, will contain a list of all records that matched the search criteria.

Signature:
Public Function search(ByVal iEntityType as Integer, _

rsSearch as ADODB.Recordset
Optional sName as String, _
Optional sFName as String, _
Optional sLName as String, _
Optional sCity as String, _
Optional sZip as String) as Long

Parameters:
iEntityType: Specifies the type of entity to be searched. shame: Specifies the organization name of the entity to be searched. This valid when searching on any entity except doctors.
sFName: Specifies the first name of the entity to be searched. This is only valid when searching for doctors.
sLName: Specifies the last name of the entity to be searched. This is only valid when searching for doctors.
sCity: Specifies the city of the entity to be searched.
sZip: Specifies the zip code of the entity to be searched.
rsSearch: An ADO Recordset which will contain all matches for the search.

The following columns are returned: name, entity_id, entity_type, fNarne, LName, City, StateCode, zip.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getErrorMsg

Description:
This method will select the Error Message from the database and return it to the calling program. The return value is a string; which can be displayed to the user in the form of a message/alert window.

Signature:
Public Function getErrorMsg(ByVal lErrorCode, _
rsErrorMsg as ADODB.Recordset) as Long

Parameters:
lErrorCode: Specifies the error in the database.
rsErrorMsg: Recordset to contain the returned error data.

The following columns are returned: code_id, code_description

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getCodeTable

Description:

This method will return all records from the specified code table.

Signature:
Public Function getCodeTable(sCodeTable as String, _
rsCodeTable as ADODB.Recordset)

Parameters:
sCodeTable: Specifies the code table to be returned.
rsCodeTable: Specifies a recordset containing the ids and names of the fields in the code table specified.

The following columns are returned: Field_Id, Field_Name.

Return type:
Integer: returns positive if successful and 0 or negative if unsuccessful.

getDrugs

Description:
The two dimensional array or ADODB.Recordset returned will contain the specified entity's drug list.

Signature:
Public Function getDrugs(ByVal iEnityId as Integer, _
ByVal iEntityType as Integer, _
rsDrugs as ADODB.Recordset) as Long

Parameters:
iEntityId: Specifies the entity for whom the drug list is requested.
iEntityTYpe: Specifies the entity type of the entity for whom the drug list is requested.
rsDrugs: An ADO Recordset which will contain all of the prescription records linked to a patient.

The following columns are returned: prescription_id, description, doctor_id, isActive.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateDrugs

Description:
This method updates the progress_prescription table with the start_date, end_date.

Signature:
Public Function updateDrugs(ByVal iPrescriptionId as Integer, _
sStartDate as String, _
sEndDate as String
Optional ByVal iPharmacistId as Integer, -

Optional isActive as Boolean) as Long

Parameters:
iPrescriptionId: Specifies the drug record being updated
sStartDate: Specifies the prescription starting date.
sEndDate: Specifies the prescription ending date.
iPharmacyId: Specifies the pharmacy for whom the drug list is requested.
isActive: Specifies whether the drug is actively used.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getProblems
The two dimensional array or ADODB.Recordset returned will contain the specified entity's problem(diagnosis) list.

Signature:
Public Function getProblems(ByVal iEntityId as Integer, _
ByVal iEntityType as Integer, _
rsProblems as ADODB.Recordset) as Long

Parameters:
iEntityId: Specifies the enitity for whom the problem list is requested.
iEntityTYpe: Specifies the type of the entity for whom the problem list is requested.
rsProblems: An ADO Recordset which will contain all of the problem records linked to a patient.

The following columns are returned: problem_id, description, doctor_id, isActive.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateProblems

Description:
This method updates the progress_problem table with the active flag.

Signature:
Public Function updateProblem(ByVal iProblemId as Integer, _
isActive as Boolean) as Long

Parameters:
iProblemId: Specifies the problem being updated.
isActive: Determines whether the problem is actively used.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getEntitv

Description:
The two dimensional array or ADODB.Recordset returned will contain the fields in the entity table corresponding to the entity and entity type passed.

Signature:
Public Function getEntity(ByVal iEntityId as Integer, _
ByVal iEntityType as Integer, _
rsEntity as ADODB.Recordset) as Long

Parameters:
iEntityId: Specifies the entity requested.
iEntitytype: Specifies the type of entity requested.
rsEntity: An ADO Recordset which will contain the referenced entity.

The following columns are returned: all fields from the specified entityType table.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateEntity

Description:
This method is used to insert or update an entity record.

Signature:
Public Function updateEntity(Byval iEntityId as Integer, _
ByVal iEntityType as Integer, _
Optional sFName as String, -
Optional sLName as String, _
Optional sOrgName as String, -
Optional sDOB as String, _
Optional sSSN_FEI as String, _
Optional sUPIN_Lic as String, _
Optional iDoctor as Integer) as Long

Parameters:
iEntityId: Specifies the entity whose record is being updated. If iEntityId = 0 the it specifies that the entity is to be inserted.
iEntityType: Specifies the entity type of the entity whose record is being updated/inserted.
sFName: Specifies the first name of the entity being inserted/updated.
sLName: Specifies the last name of the entity being inserted/updated.
sOrgName: Specifies the organization name of the entity being inserted/updated.
sDOB: Specifies the DOB of the entity being inserted/updated.
sSSN_FEI: Specifies the SSN or FEI of the entity being inserted/updated.

sUPIN_Lic: Specifies the UPIN or License of the entity being inserted/updated. License for pharmacy, UPIN for doctor, license for Pharmacy.
iDoctor: Specifies the parent doctor when inserting a doctor. Specifies the doctor for a patient when inserting a doctor record.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getPatientInsurance

Description:
The two dimensional array or ADODB.Recordset returned will contain the insurance information for the specified patient. Multiple records may be returned where each record

Signature:
Public Function get Patient Insurance (ByVal iPatientId as Integer, _
rsInsurance as ADODB.Recordset) as Long

Parameters:
iPatientId: Specifies the patient whose insurance is requested.
rsInsurance: An ADO Recordset which will contain the patient's insurance information. Each row will contain an insurance record for the patient.

The following columns are returned: insurance_id, policy, name, FEI

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updatePatientInsurance

Description:
Adds or updates an Insurance record for an existing patient.

Signature:
Public Function updatePatientInsurance(ByVal iPatientId as Integer, _
ByVal iInsurance Id as Integer, -
Optional sPolicy as String, _
Optional sName as String, _
Optional sFEI as String) as Long

Parameters:
iPatientId: Specifies the patient for whom the insurance information is requested.
iInsuranceId: Specifies the Insurance company inserted/updated for the patient. If the value for iInsurance_Id = 0, then this method will insert a new insurance company record.
sPolicy: Specifies the policy number for the insurance company and patient.
sName: Specifies the name of the Insurance company that is being inserted/updated
sSFEI: Specifies the FEI for the Insurance company being inserted/updated.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getAddressInfo

Description:
The two dimensional array. or ADODB.Recordset returned will contain the address information for the entity being passed. Each record will represent one address type for the entity. An entity may have 0 to many address records, but each one will have a different address type.

Signature:
Public Function getAddressInfo(ByVal iEntityId as Integer, _
ByVal iEntityTypeId as Integer, _
rsAddressInfo as ADODB.Recordset) as Long

Parameters:
iEntityId: Specifies the entity whose address is requested.
iEntityTypeId: Specifies the entitytype of the entity whose address is requested.
rsAddressinfo: An ADO Recordset which will contain the referenced entities address records.

The following columns are returned: name, addressType_id, address1, address2, city, stateCode, zip, country.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateAddressinfo

Description:
This method will insert or update an address record for the specified entity. If the address type parameter passed is not equal to the address type field in the address table for that entity and entity type, then a new address record will be created. If not, the address record identified by the three passed fields will be updated.

Signature:
Public Function updateAddressInfo(
ByVal iEntityId as Integer, _
ByVal iEntityTypeId as Integer, _
ByVal iAddressTypeId as Integer, _
Optional sAddress1 as String, _
Optional sAddress2 as String, _
Optional sCity as String,
Optional sStateCode as String, _
Optional sZip as String, _
Optional sCountry as String) as Long

Parameters:

iEntityId: Specifies the entity id of the entity whose address is being inserted/updated.
iEntityTypeId: Specifies the type of entity whose address is being inserted/updated.
iAddressTypeId: Specifies the address type of the entity whose address is being inserted/updated. If this value is not equal to an existing address record, then the address record is an insert. If not it is an update.
sAddress1 : Specifies the first line of the address of the entity whose address is being inserted/updated.
sAddress2: Specifies the second line of the address of the entity whose address is being inserted/updated.
sCity: Specifies the city of the entity whose address is being inserted/updated.
sStateCode: Specifies the state of the entity whose address is being inserted/updated.
sZip: Specifies the postal code of the entity whose address is being inserted/updated.
sCountry: Specifies the country of the entity whose address is being inserted/updated.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getContactInfo

Description:
The two dimensional array or ADODB.Recordset returned will contain the contact information for the entity being passed. Each record will represent one contact type for the entity. An entity may have 0 to many contact records, but each one will have a different contact type.

Signature:
```
Public Function getContactInfo(ByVal iEntityId as Integer, _
                               ByVal iEntityTypeId as Integer, _
                               rsContactInfo as ADODB.Recordset) as Long
```

Parameters:
iEntityId: Specifies the entity whose contact info is requested.
iEntityTypeId: Species the entitytype of the entity whose contact info is requested.
rsContactInfo: An ADO Recordset which will contain the referenced entities contact information.

The following columns are returned: contact_type, value.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateContactInfo

Description:
This method will insert or update a contact record for the specified entity. If the contact type parameter passed is not equal to the contact type field in the contact table for that entity and entity type, then a new contact record will be created. If not, the contact record identified by the three passed fields will be updated.

Signature:
Public Function updateContactInfo(
ByVal iEntityId as Integer, _
ByVal iEntityTypeId as Integer, _
ByVal iContactTypeId as Integer, -
Optional sValue as String) as Long

Parameters:
iEntityId: Specifies the entity id of the entity whose address is being inserted/updated.
iEnftyTypeId: Specifies the type of entity whose address is being inserted/updated.
iContactTypeId: Specifies the address type of the entity whose address is being inserted/updated. If this value is 0 then the address record is an insert. If not it is an update.
sValue: Specifies the value of the contact type for the entity whose contact is being inserted/updated.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

bill

Description:
This method is called after a patient's record is accessed, where the patient's doctor_id is not equal to the doctor_id of the current user, or the current user is not a doctor. This method is called after a progress note record is accessed where the progress note's doctor_id is not equal to the doctor_id of the current user, or the current user is not a doctor.

Signature:
Public Function bill(ByVal iEntityId as Integer, _ _
ByVal iEntityType as Integer, _
ByVal iTransactionType as Integer) as Long

Parameters:
iEntityId: Specifies an array of entity id of the entities being billed.
iEntityTypeId: Specifies an array of entityType ids of the entity being billed.
iTransactionType: Specifies the transaction type; patient access, note access, pharmaceutical report.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

get Bills

Description:
This method will first get all bill records that are over 30 days old and do not have a receipt record. An AR transaction of type, aged receivables, will be created in the amount of 10% of the bill amount. The bill will then be updated with the new amount, based on the original bill

amount plus the new transaction amount. This method will then get all AR records that do not have a bill (batch) number, group them by entityId and entityType, and sum them on amount. These groups will then be inserted as new bill records. Each of these bill records, including all newly created and all updated bill will be returned. If the bDetail flag is true, all corresponding AR records that make up each bill will also be returned

Signature:
Public Function getBills(rsBills as ADODB.Recordset
bDetail as Boolean) as Long

Parameters:
rsBills: Contains a list of Bill records, grouped by entity_ID and entity-type

The following columns are returned: EntityId, EntitytypeID, EntityFName, EntityLName, EntityOrgName, Bill_Date, Amount, Tran_Date, Tran_Type, Acct_Rec.Amount.

bDetail: Specifies whether the returned recordset will contain A/R records or just bill (summary) records.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getAcctsPayable

Description:
This method will return all AP records that do not have a check (batch) number.

Signature:
Public Function getAcctsPayable(rsAP as ADODB.Recordset) as Long

Parameters:
rsAP: Contains a list of AP records, grouped by entity_ID and entity_type

The following columns are returned: EntityId, EnftytypeID, EntityFName, EnftyLName, EntityOrgName, Tran_Date, Tran_Amount.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

processChecks

Description:
This method will create a check record and update the AP record with the batch number of the created check.

Signature:
Public Function processChecks(iEntityId() as Integer, _

iEntityTypeId() as Integer, _
vCheckDate() as Variant, _
iCheckNumber() as Integer, _
vCheckAmount() as Variant) as Long

Parameters:
iEntityId(): Specifies an array of entity id of the entities whose check records are being updated.
iEntityTypeId(): Specifies an array of entityType ids of the entities whose check records are being updated.
vCheckDate: Specifies an array of the dates of the check records being updated.
ICheckNumber: Specifies an array of the check numbers of the check records being updated.
vCheckAmount: Specifies an array of the amounts of the check records being updated.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

processReceipts

Description:
This method will update the Payment Received table with payments amounts and dates for the corresponding bills.

Signature:
Public Function processReceipts(iEntityId() as Integer, _
iEntityTypeId() as Integer, _
vRecDate() as Variant, _
vRecAmount() as Variant) as Long

Parameters:
iEntityIdo: Specifies an array of entity id of the entities whose bills are being paid.
iEntityTypeId(): Specifies an array of entityType ids of the entities whose bills are being paid.
vRecDate: Specifies an array of the dates of the receipts for the bills that were paid.
vRecAmount: Specifies an array of the amounts of the receipts for the bills that were paid.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getReceipts

Description:
This method will return all Payment Received records where there is no Deposit id.

Signature:
Public Function getReceipts(rsReceipts as ADODB.Recordset) as Long

Parameters:
rsReceipts: Contains a list of Payment Received records.

The following columns are returned: EntityId, EntitytypeID, EntityFName, EnftyLName, EnftyOrgName, ReceiptID, Receipt_Date,, Amount.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateDeposit

Description:
This method will create a deposit record and update the receipt records included on that deposit with the deposit id.

Signature:
Public Function updateDeposit(iReceiptId() as Integer, _
sDepDate as String, _
sDepTicketId as String) as Long

Parameters:
iReceiptId: Specifies an array of Receipts to be updated with the deposit id.
sDepDate: Specifies the date of the deposit.
sDepTicketId: Specifies the deposit ticket id for the deposit.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

getReconciliation

Description:
This method will return all deposits records with no clear_date and all checks with no check_clear_date.

Signature:
Public Function getReconciliation(rsChecks as ADODB.Recordset, _
rsDeposits as ADODB.Recordset) as Long

Parameters:
rsChecks: Contains check records where there is no check-clear-date for that check.

The following columns are returned: Entity_Id, Entity_Type, Check_Date, Check_Number, Check_Amount.

rsDeposits: Contains deposit records where there is no clear date for that deposit.

The following columns are returned: Entity_Id, Entity_Type, Deposit_Date, Deposit_Amount.

Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

updateReconciliation

Description:
This method will update the checks table with a check_clear_date for checks that clear and updates the deposit table with a clear_date for deposits that clear.

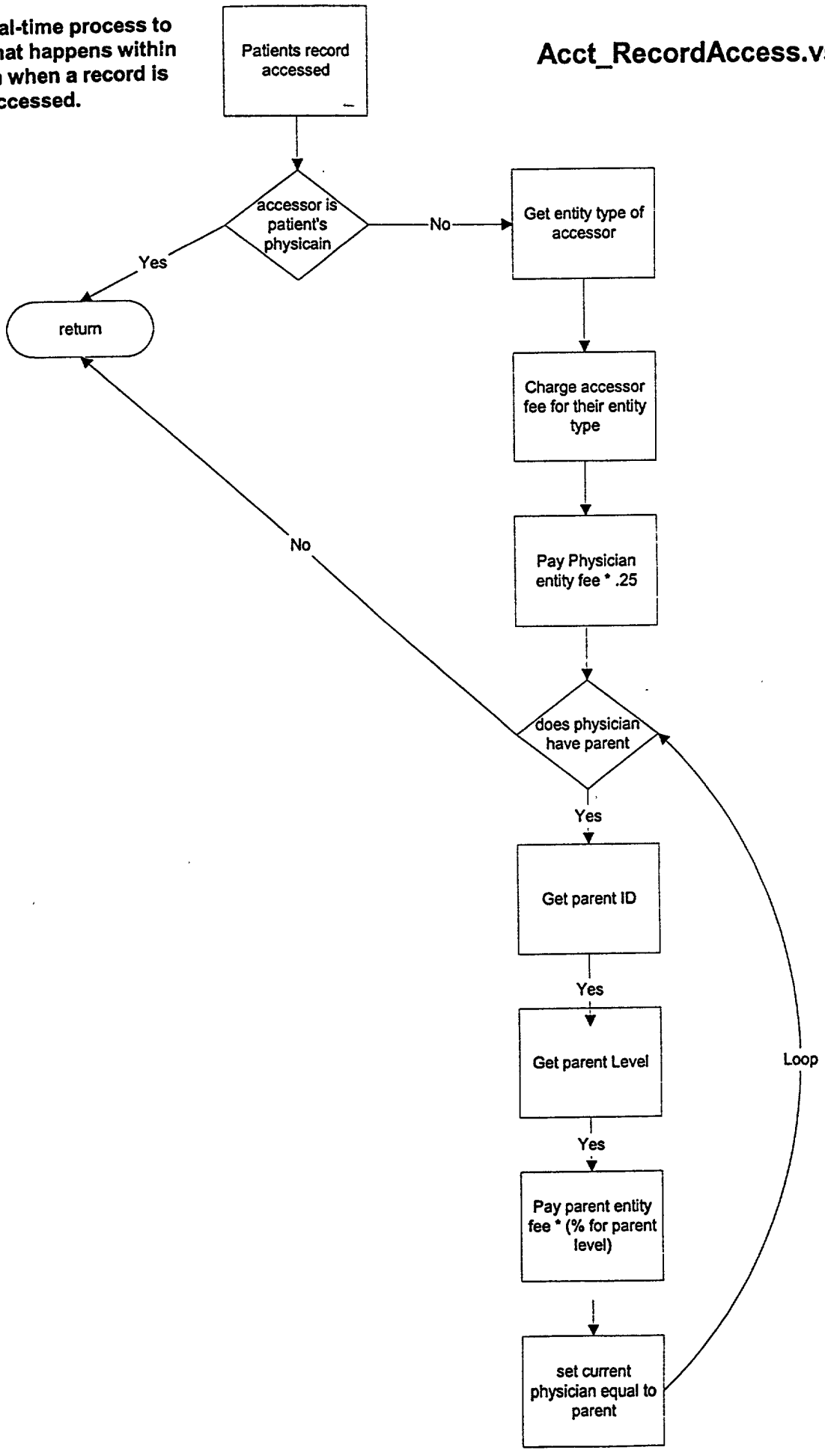
Signature:
Public Function updateReconciliation(iBatch() as Integer, _
vCheckAmount() as Variant, _
sCheckDate() as String, _
iDepositID() as Integer, _
vDepositAmount() as Variant, _
sDepositDate() as String) as Long

Parameters:
iBatch(): Array of check batch numbers. Each item in the array corresponds to a check amount and dates in the vCheckAmount array and the sCheckDate array.
vCheckAmount(): Specifies an array of check amounts being reconciled.
sCheckClearDate(): Specifies an array of the dates those checks cleared on the bank statement.
iDepositId(): Specifies an array of deposit ids being reconciled.
vDepositAmount(): Specifies an array of the deposit Amounts being reconciled.
sDepositClearDate(): Specifies an array of the dates those deposits cleared on the bank statement.

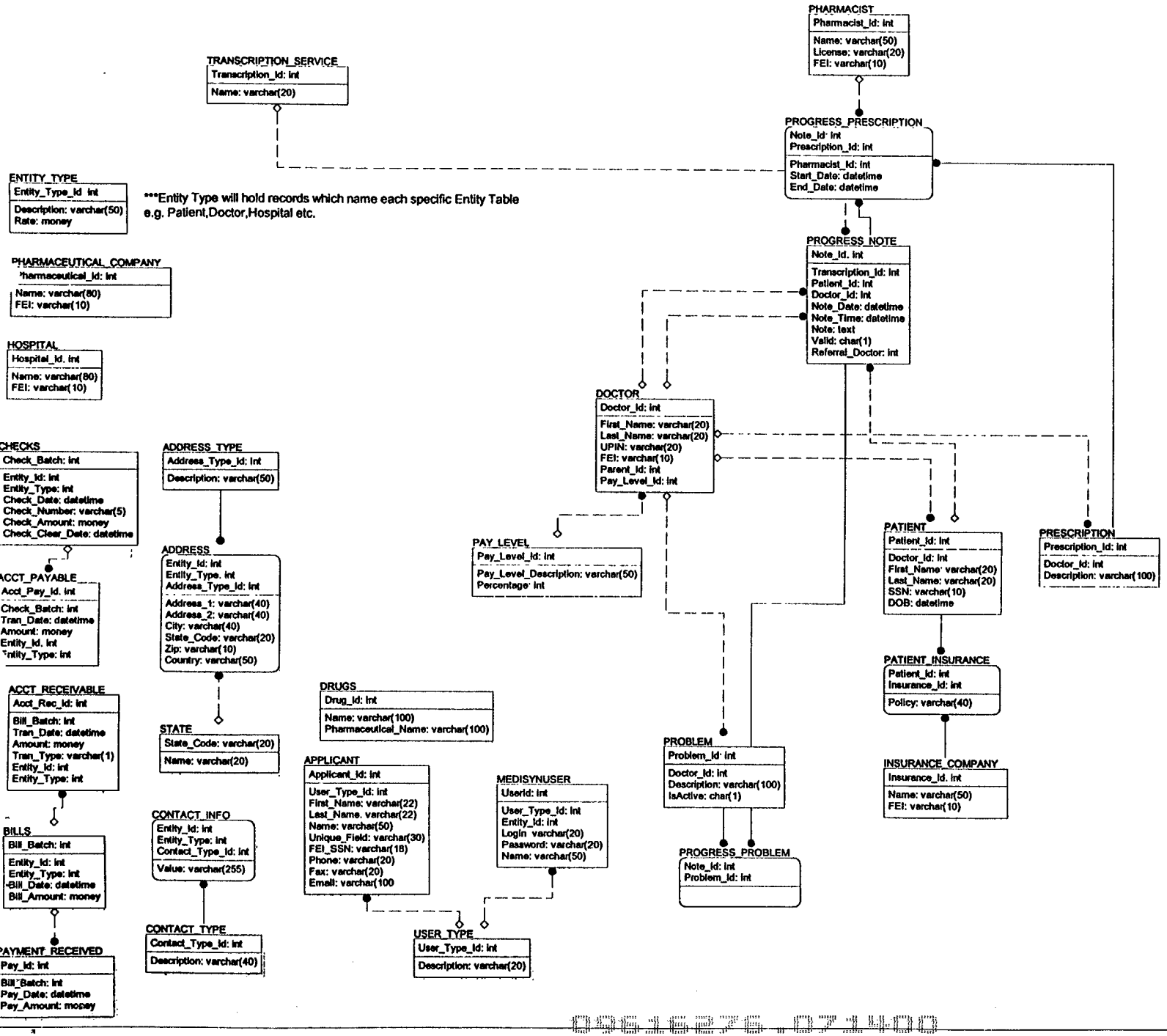
Return type:
Long: returns 0 if successful and an Error Code if unsuccessful.

This is a real-time process to describe what happens within the system when a record is accessed.

Acct_RecordAccess.vsd



MEDISYN DATA MODEL



Data Dictionary

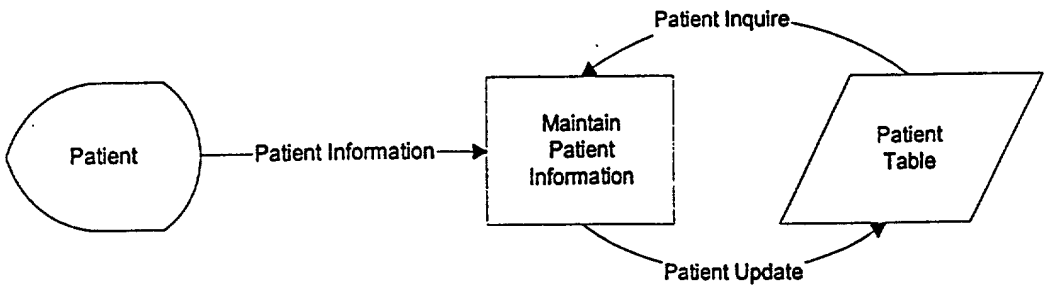
Entity Name	Entity Attribute Name	Entity Attribute Definition	Entity Attribute Column Datatype
ACCT_PAYABLE	Acct_Pay_Id	Unique sequential number referencing each record in the table	int
	Check_Batch	The specific check that this transaction is associated with. Must be valid in the CHECKS table.	
	Tran_Date	Date of account payable transaction	datetime
	Amount	Amount of account payable transaction	money
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	int
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	
ACCT_RECEIVABLE	Acct_Rec_Id	Unique sequential number referencing each record in the table	
	Bill_Batch	The specific bill that this transaction is associated with. Must be valid in the BILLS table.	
	Tran_Date	Date of account receivable transaction	datetime
	Amount	amount of account payable transaction	money
	Tran_Type	Type of account receivable transaction. Valid values are: O - Overpaid, U - Underpaid, R - Record Access	varchar(1)
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	int
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	
ADDRESS	City	The applicable city.	varchar(40)
	Address_2	The second line of the street address.	
	State_Code	Code identifyingstate Must be valid in STATE table.	varchar(20)
	Zip	The standard postal zip code.	varchar(10)
	Country	Name of Country.	varchar(50)
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	int
	Address_1	The first line of the street address.	varchar(40)
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	int
	Address_Type_Id	Code identifying type of Address. Must be valid in ADDRESS_TYPE table.	
ADDRESS_TYPE		Unique sequential number referencing each record in the table	
	Description	The textual description identifying a specific type of address.	varchar(50)
BILLS	Bill_Batch	Unique sequential number referencing each record in the table	int
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	
	Bill_Date	Date Bill was created.	datetime
	Bill_Amount	Amount of Bill.	money
CHECKS	Check_Batch	Unique sequential number referencing each record in the table	int
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	
	Check_Date	Date Check was created	datetime
	Check_Number	Number of Check	varchar(5)

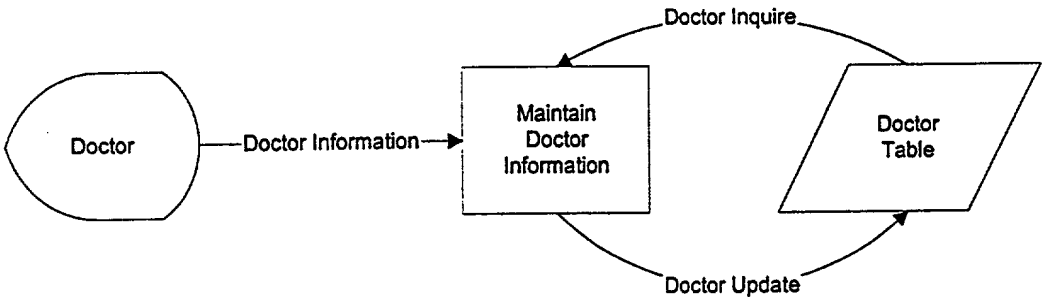
Entity Name	Entity Attribute Name	Entity Attribute Definition	Entity Attribute Column Datatype
CHECKS	Check_Amount	Amount of Check	money
	Check_Clear_Date	Date check cleared bank.	datetime
CONTACT_INFO	Contact_Type_Id	Code identifying type of Address. Must be valid in CONTACT_TYPE table.	int
	Value	Text field containing contact information	varchar(255)
	Entity_Type	Identifies type of entity. Must be valid in ENTITYTYPE table.	int
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	
CONTACT_TYPE	Contact_Type_Id	Unique sequential number referencing each record in the table	
	Description	The textual description identifying a specific type of contact	varchar(40)
DOCTOR	FEI	Federal Identification Number	varchar(10)
	UPIN	UPIN Number of Doctor	varchar(20)
	Parent_Id	Identifier of physician which recruited doctor. Pointer to DOCTOR Table.	int
	Pay_Level_Id	Code representing the level of pay out for the doctor. Must be valid in the PAY_LEVEL table.	
	Doctor_Id	Unique sequential number referencing each record in the table	
	Last_Name	Last Name of Doctor	varchar(20)
	First_Name	First Name of Doctor	
ENTITY_TYPE	Entity_Type_Id	Unique sequential number referencing each record in the table	int
	Description	The textual description identifying a specific type of entity.	varchar(50)
	Rate	Rate of pay for specific entity.	money
HOSPITAL	Hospital_Id	Unique sequential number referencing each record in the table	int
	Name	Name of Hospital	varchar(80)
	FEI	Federal Identification Number	varchar(10)
INSURANCE_COMPANY	Insurance_Id	Unique sequential number referencing each record in the table	int
	Name	Name of Insurance Company	varchar(50)
	FEI	Federal Identification Number	varchar(10)
MEDISYNUSER	UserId	User ID of system user	int
	Type	Type of User	varchar(10)
	Entity_Id	Identifies the unique record in associated EntityTable. Must be valid in associated entity table.	int
	Login	Login of User	varchar(20)
	Password	Password of User	
	Name	Name of System User if not in any Entity Table.	varchar(50)
PATIENT	Patient_Id	Unique sequential number referencing each record in the table	int
	First_Name	First Name of Patient	varchar(20)
	Last_Name	Last Name of Patient	
	SSN	Patients Social Security Number	varchar(10)
	DOB	Patients Date of Birth	datetime
PATIENT_INSURANCE	Patient_Id	Identifies the associated patient that this record belongs to. Must be valid in PATIENT table.	int
	Insurance_Id	Identifies the associated insurance that this record belongs to. Must be valid in INSURANCE table.	
	Policy	Number of Insurance Policy	varchar(40)
PAY_LEVEL	Pay_Level_Id	Unique sequential number referencing each record in the table	int
	Pay_Level_Description	The textual description identifying a specific pay level.	varchar(50)

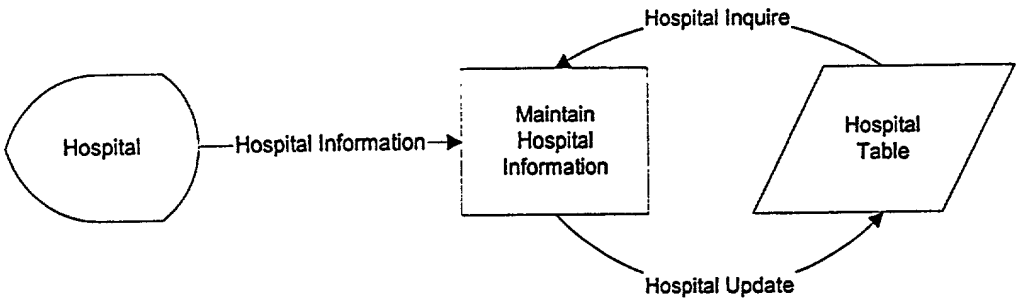
Entity Name	Entity Attribute Name	Entity Attribute Definition	Entity Attribute Column Datatype
PAY_LEVEL	Percentage	Percentage of Payout.	int
PAYMENT_RECEIVED	Pay_Date	Date of payment	datetime
	Pay_Amount	Amount of payment	money
	Bill_Batch	Identifies the associated bill that this payment belongs with.. Must be valid in BILLS table.	int
	Pay_Id	Unique sequential number referencing each record in the table	
PHARMACEUTICAL_COMPANY	Pharmaceutical_Id	Unique sequential number referencing each record in the table	
	Name	Name of Pharmaceutical Company	varchar(80)
	FEI	Federal Identification Number	varchar(10)
PHARMACIST	Pharmacist_Id	Unique sequential number referencing each record in the table	int
	Name	Name of pharmacist	varchar(50)
	License	Pharmacist License Number	varchar(20)
	FEI	Federal Identification Number	varchar(10)
PRESCRIPTION	Prescription_Id	Unique sequential number referencing each record in the table	int
	Doctor_Id	Identifies the associated doctor that this prescription belongs with. Must be valid in DOCTOR table.	
	Description	The textual description identifying a specific prescription.	varchar(100)
PROBLEM	Problem_Id	Unique sequential number referencing each record in the table	int
	Doctor_Id	Identifies the associated doctor that this prescription belongs with. Must be valid in DOCTOR table.	
	Description	The textual description identifying a specific problem.	varchar(100)
	IsActive	Flag identifying whether problem is still an active problem.	char(1)
PROGRESS_NOTE	Note_Date	Date of Progress Note	datetime
	Doctor_Id	Identifies the associated doctor for this progress note. Must be valid in DOCTOR table.	int
	Note_Time	Time of Progress Note	datetime
	Note	Textual Transcription of progress note.	text
	Valid	Flag indicating whether this progress note has been deemed valid.	char(1)
	Note_Id	Unique sequential number referencing each record in the table	int
	Patient_Id	Identifies the associated patient for this progress note. Must be valid in PATIENT table.	
	Transcription_Id	Identifies the associated transcription service that input this progress note transcription. Must be valid in TRANSCRIPTION SERVICE table.	
	Referral_Doctor_Id	Identifies the associated doctor that this patient was referred to. Must be valid in REFERRAL DOCTOR table.	
PROGRESS_PRESCRIPTION	Note_Id	Identifies the associated progress note. Must be valid in PROGRESS NOTE table.	
	Prescription_Id	Identifies the associated prescription for this progress note. Must be valid in PRESCRIPTION table.	
	Pharmacist_Id	The pharmacist filling the prescription. Must be valid in PHARMACIST table.	
	Start_Date	The start date of the prescription.	datetime
	End_Date	The end date of the prescription.	
PROGRESS_PROBLEM	Note_Id	Identifies the associated progress note. Must be valid in PROGRESS NOTE table.	int

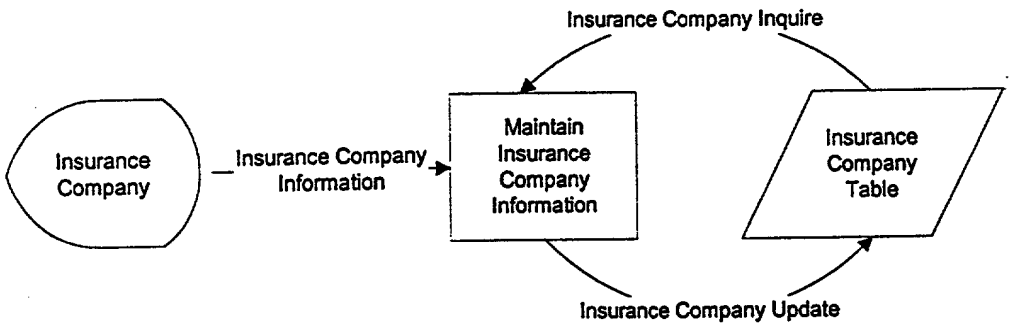
Entity Name	Entity Attribute Name	Entity Attribute Definition	Entity Attribute Column Datatype
PROGRESS_PROBLEM	Problem_Id	Identifies the associated prescription for this progress note. Must be valid in PROBLEM table.	int
REFERRAL_DOCTOR	Referral_Doctor_Id	Unique sequential number referencing each record in the table	
	First_Name	First Name of Referral Doctor	varchar(20)
	Last_Name	Last Name of Referral Doctor	
	Office_Name	Name of doctor's office	varchar(50)
	UPIN	UPIN Number	varchar(20)
STATE	State_Code	Unique code referencing each state in the table	
	Name	The textual description of a state.	
TRANSCRIPTION_SERVICE	Transcription_Id	Unique sequential number referencing each record in the table	int
	Name	Name of the Transcription Service.	varchar(20)

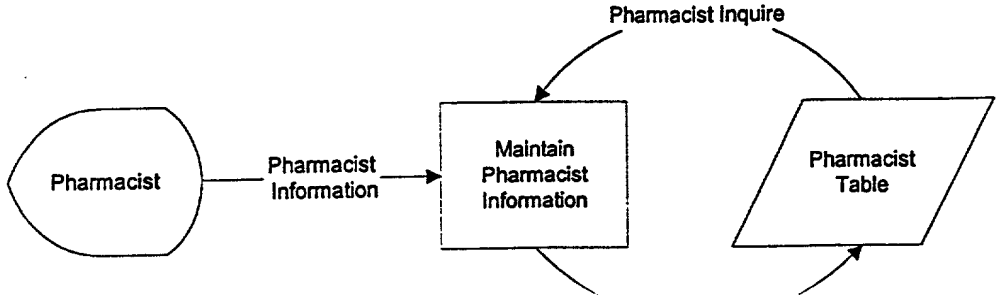
MEDISYN


Patient Inquire
Patient
Patient Information
Maintain Patient Information
Patient Table
Patient Update

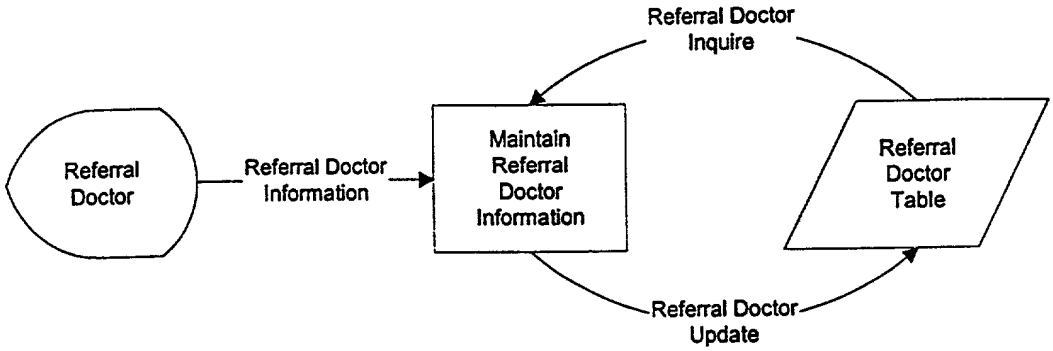

Doctor Inquire
Doctor
Doctor Information
Maintain Doctor Information
Doctor Table
Doctor Update

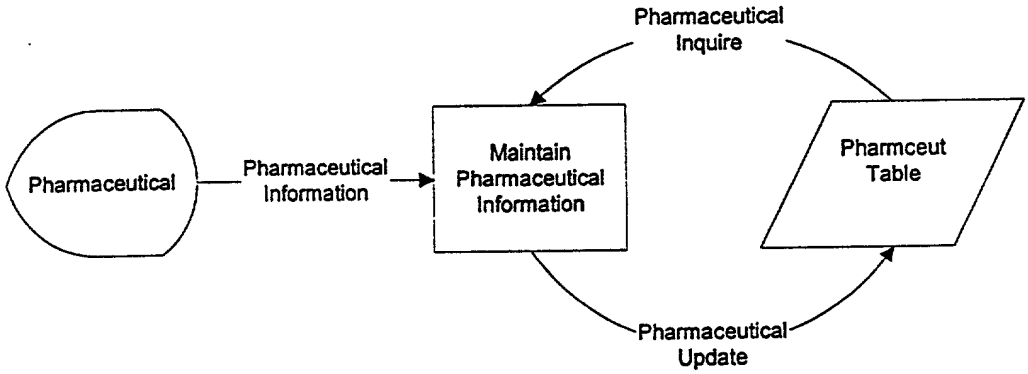

Hospital Inquire
Hospital
Hospital Information
Maintain Hospital Information
Hospital Table
Hospital Update

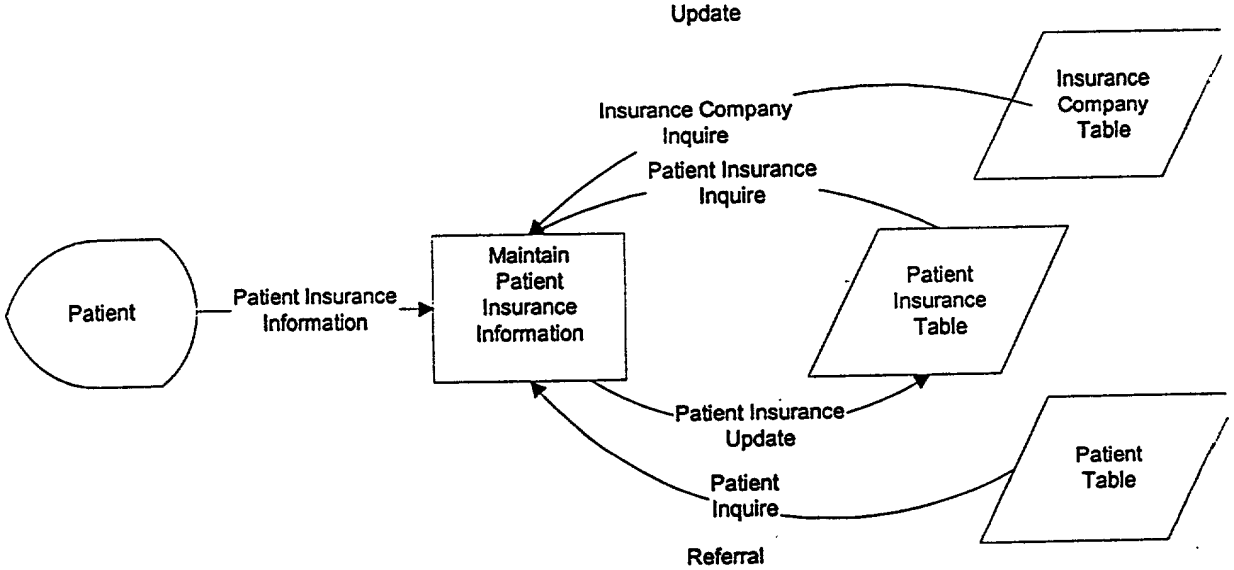

Insurance Company Inquire
Insurance Company
Insurance Company Information
Maintain Insurance Company Information
Insurance Company Table
Insurance Company Update

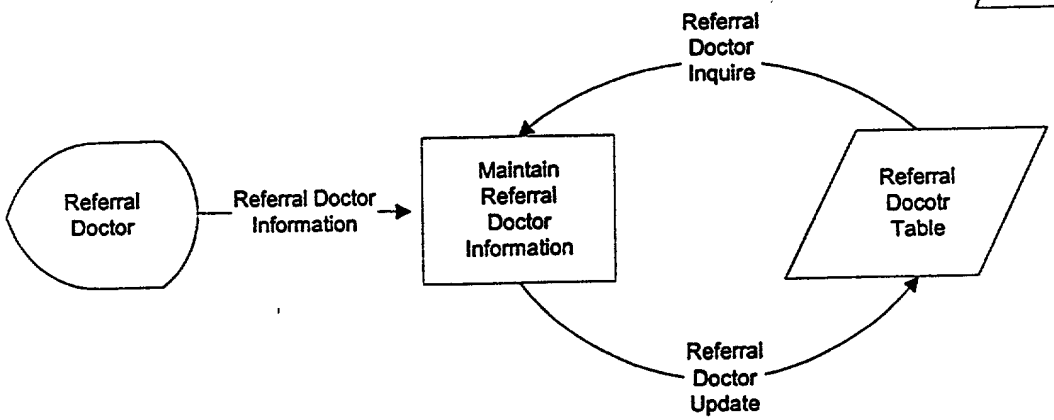

Pharmacist Inquire
Pharmacist
Pharmacist Information
Maintain Pharmacist Information
Pharmacist Table

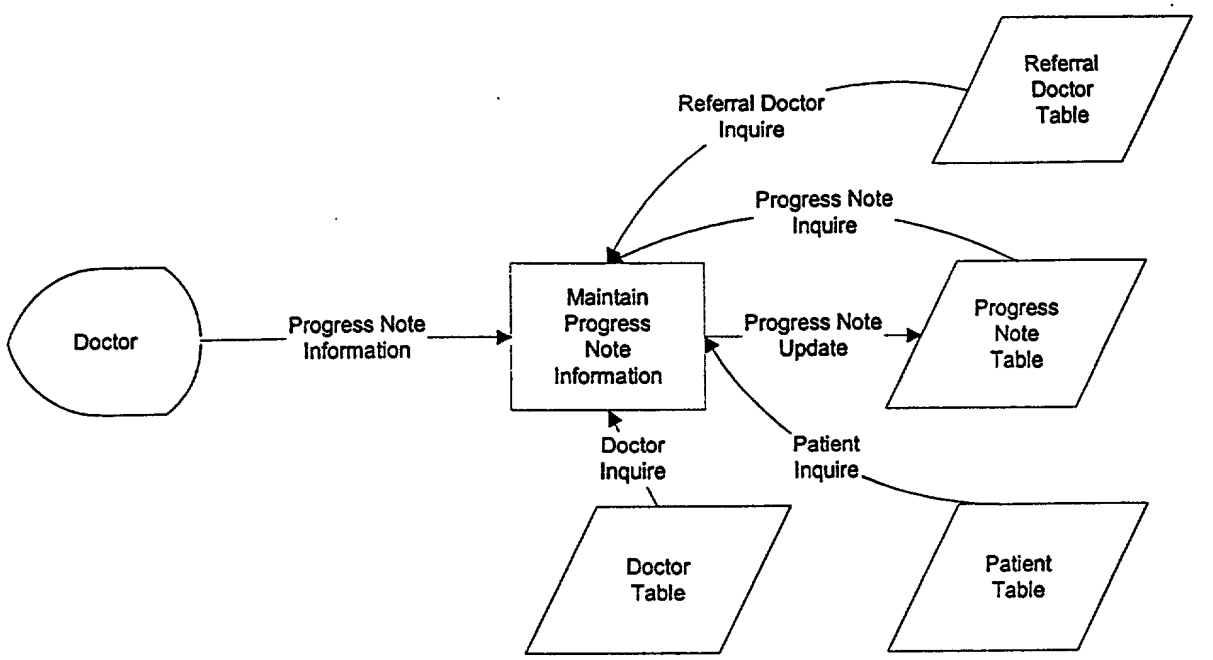
MEDISYN

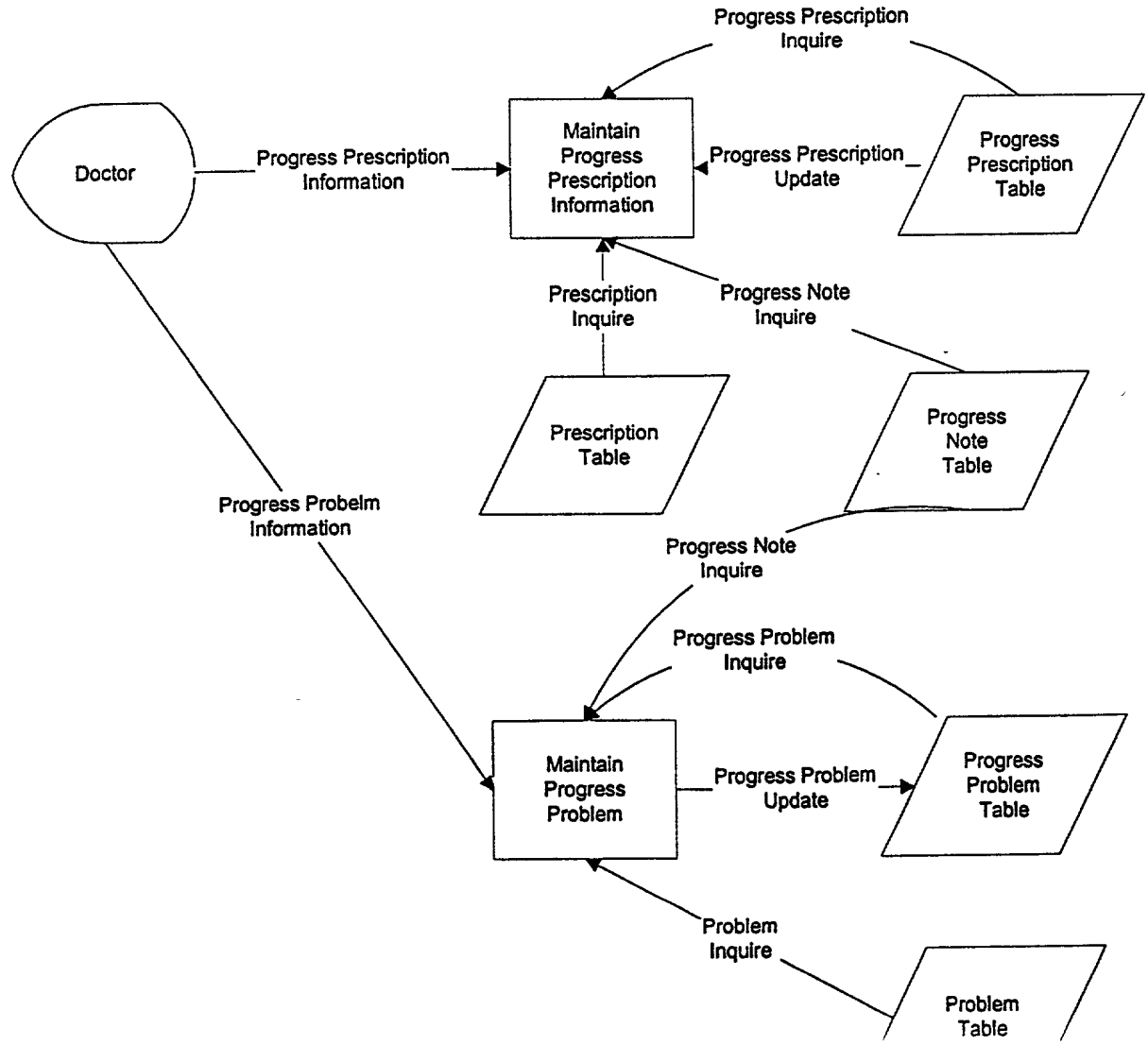


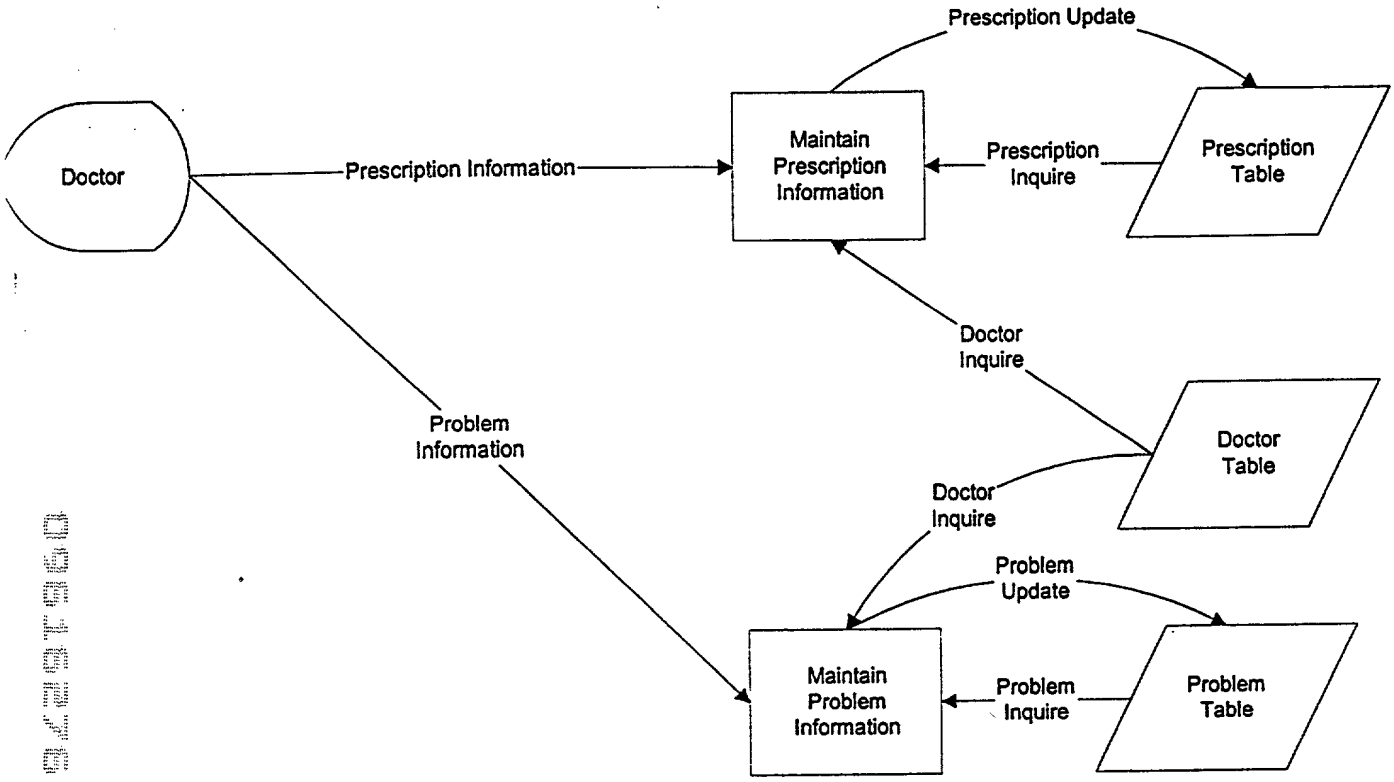




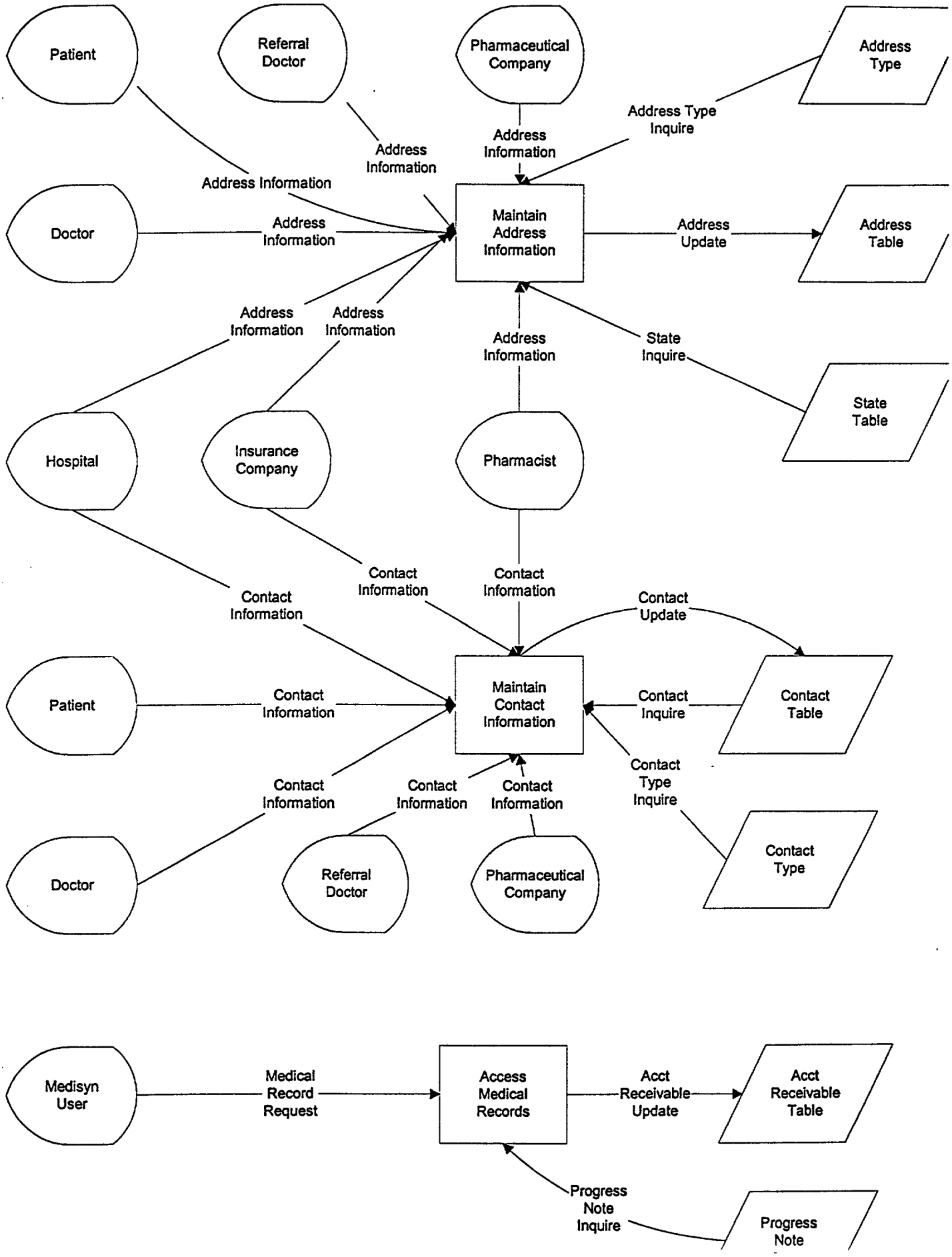


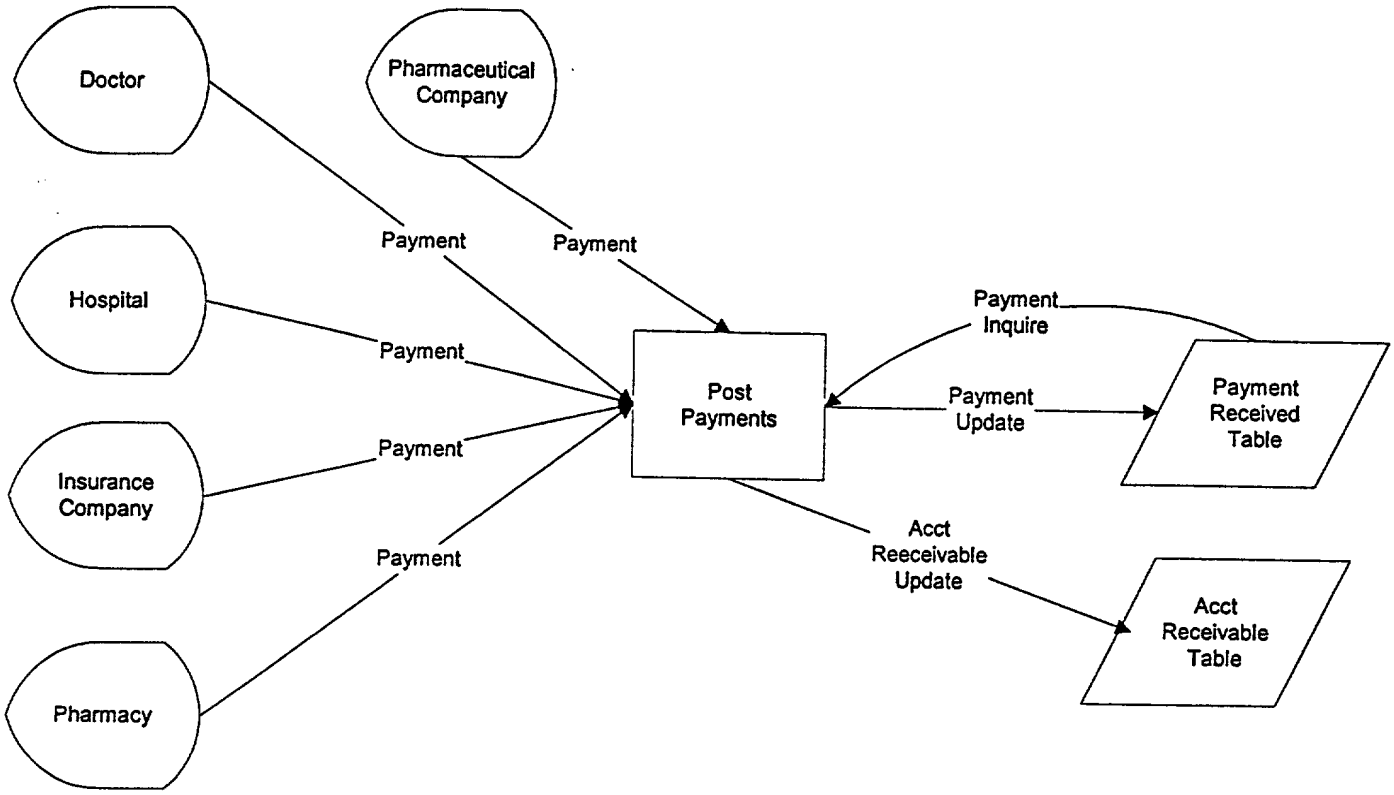

Referral Doctor Table
Referral Doctor Inquire
Progress Note Inquire
Doctor
Progress Note Information
Maintain Progress Note Information
Progress Note Update
Progress Note Table
Doctor Inquire
Patient Inquire
Doctor Table
Patient Table

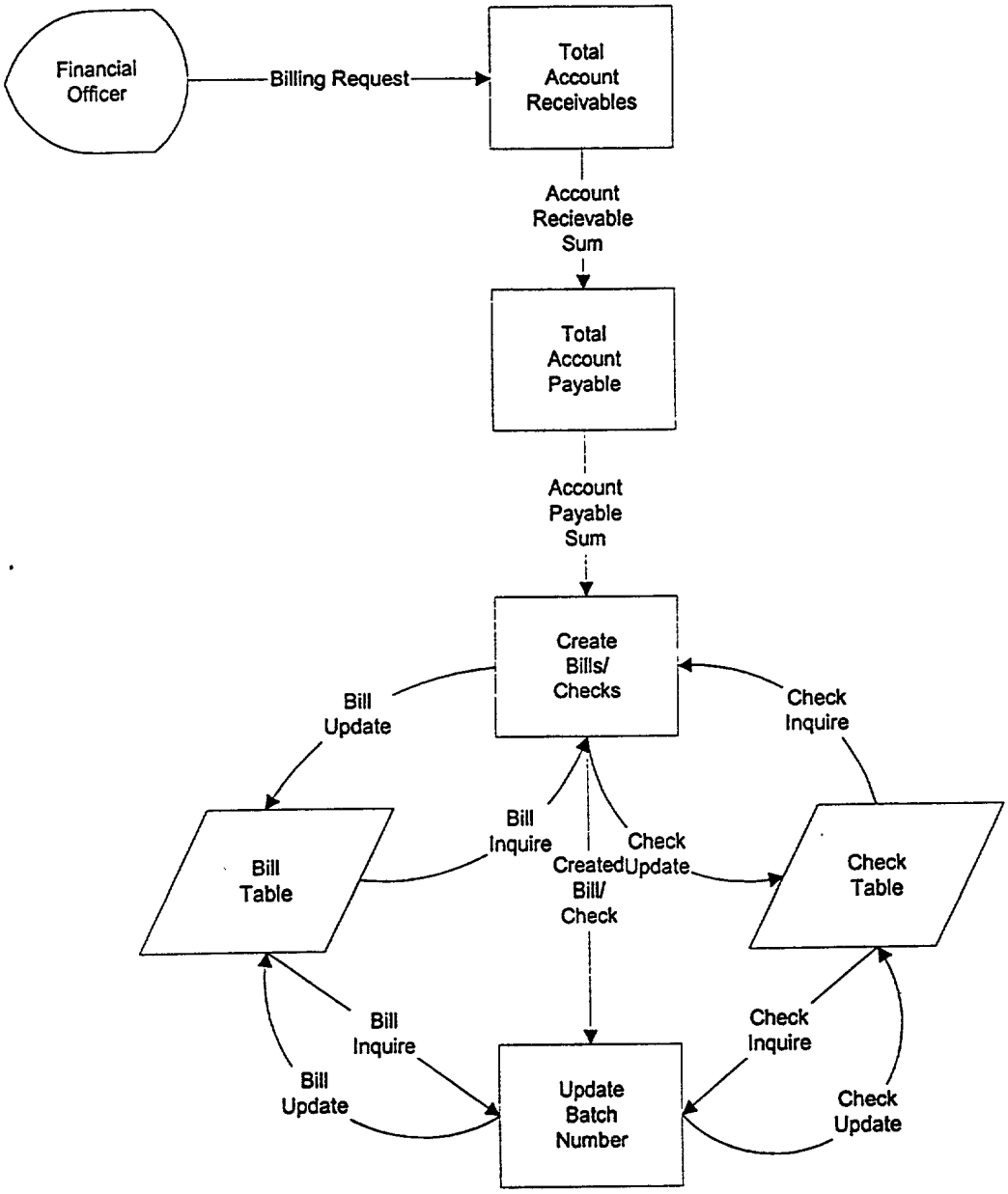

Progress Prescription Inquire
Doctor
Progress Prescription Information
Maintain Progress Prescription Information
Progress Prescription Update
Progress Prescription Table
Prescription Inquire
Progress Note Inquire
Prescription Table
Progress Note Table
Progress Probelm Information
Progress Note Inquire
Progress Problem Inquire
Maintain Progress Problem
Progress Problem Update
Progress Problem Table
Problem Inquire
Problem Table



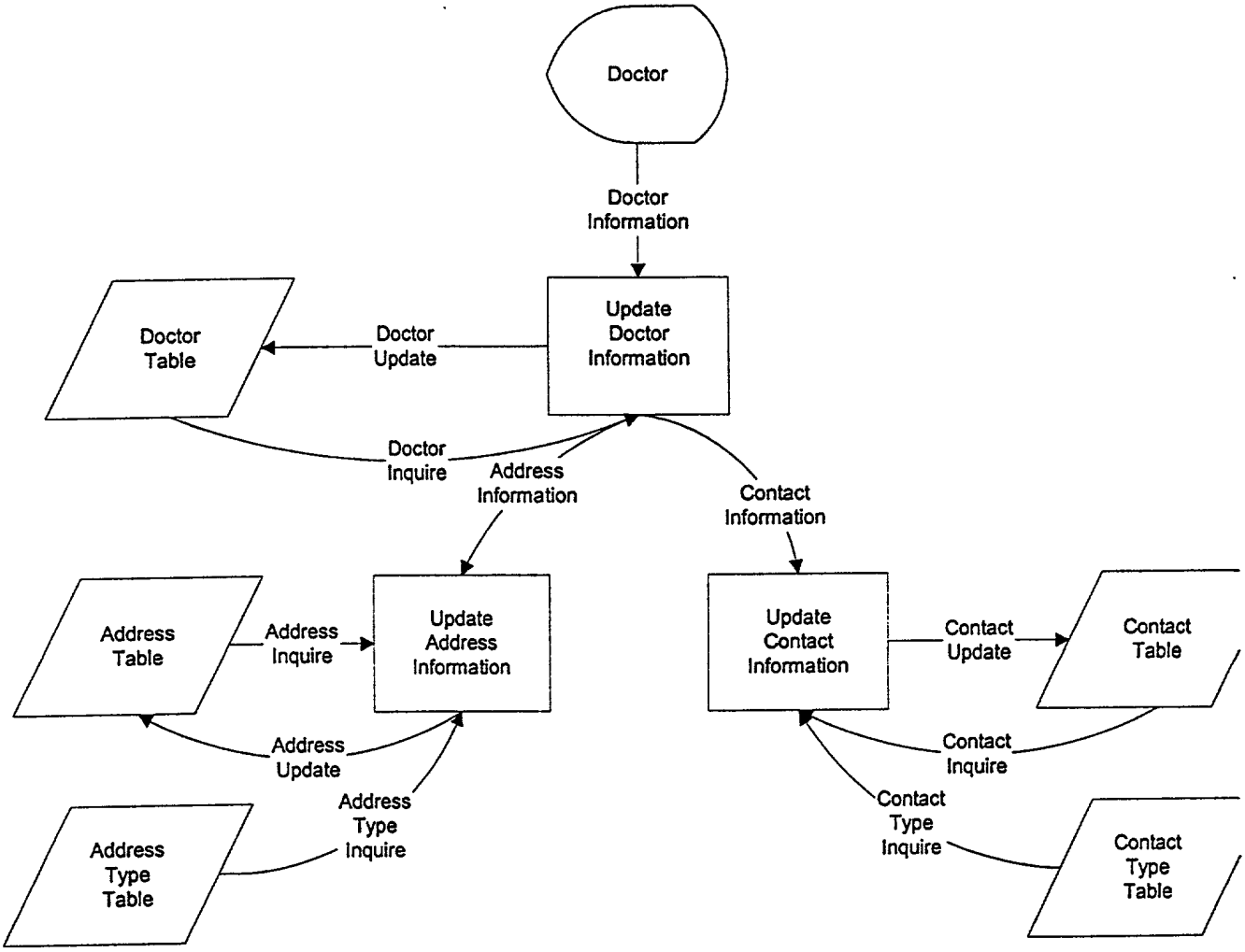
MEDISYN

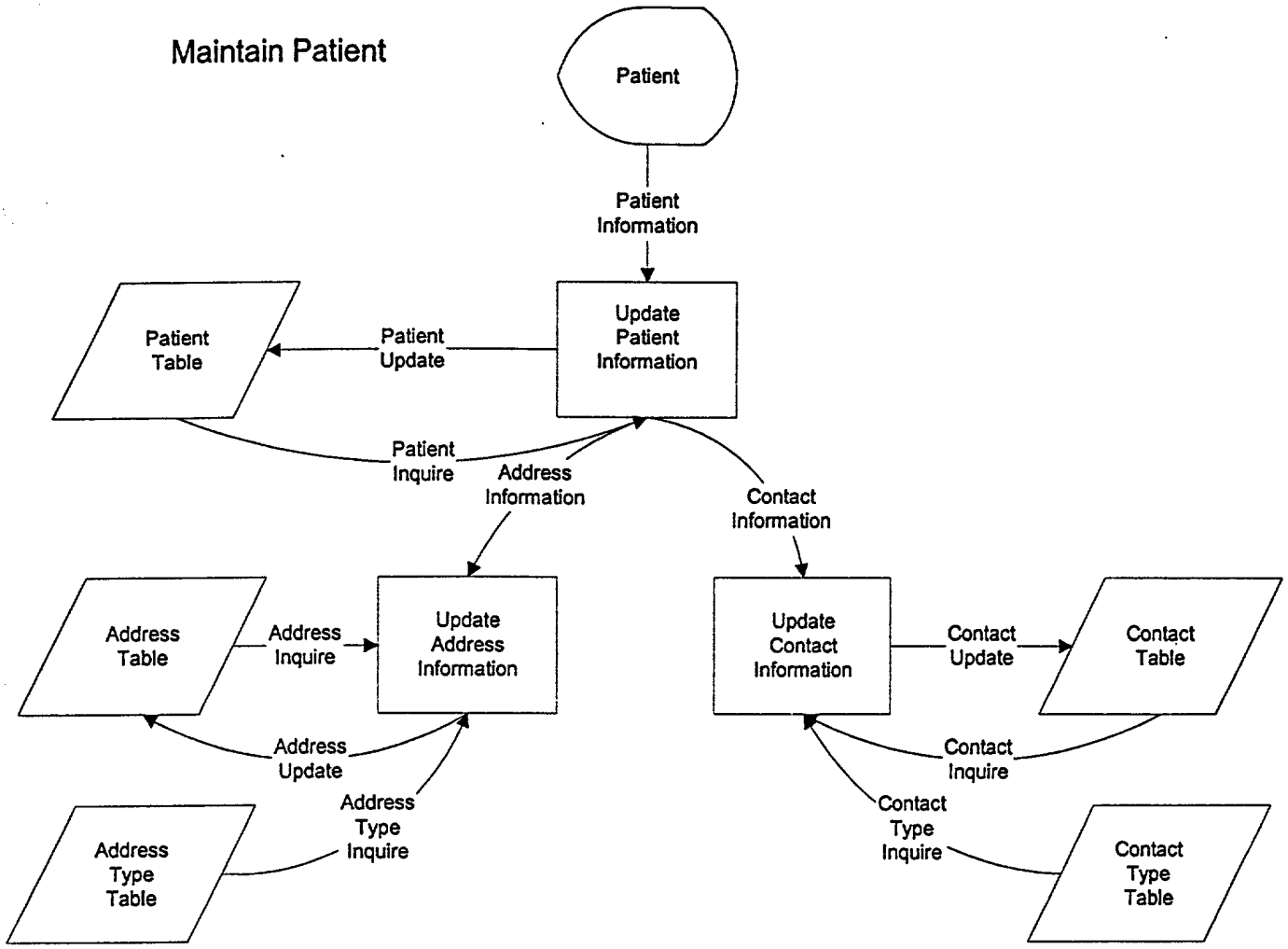

Patient
Referral Doctor
Pharmaceutical Company
Address Type
Address Type Inquire
Address Information
Address Information
Address Information
Doctor
Address Information
Maintain Address Information
Address Update
Address Table
Address Information
Address Information
Address Information
State Inquire
State Table
Hospital
Insurance Company
Pharmacist
Contact Information
Contact Information
Contact Information
Contact Update
Patient
Contact Information
Maintain Contact Information
Contact Inquire
Contact Table
Contact Information
Contact Information
Contact Information
Contact Type Inquire
Doctor
Referral Doctor
Pharmaceutical Company
Contact Type
Medisyn User
Medical Record Request
Access Medical Records
Acct Receivable Update
Acct Receivable Table
Progress Note Inquire
Progress Note

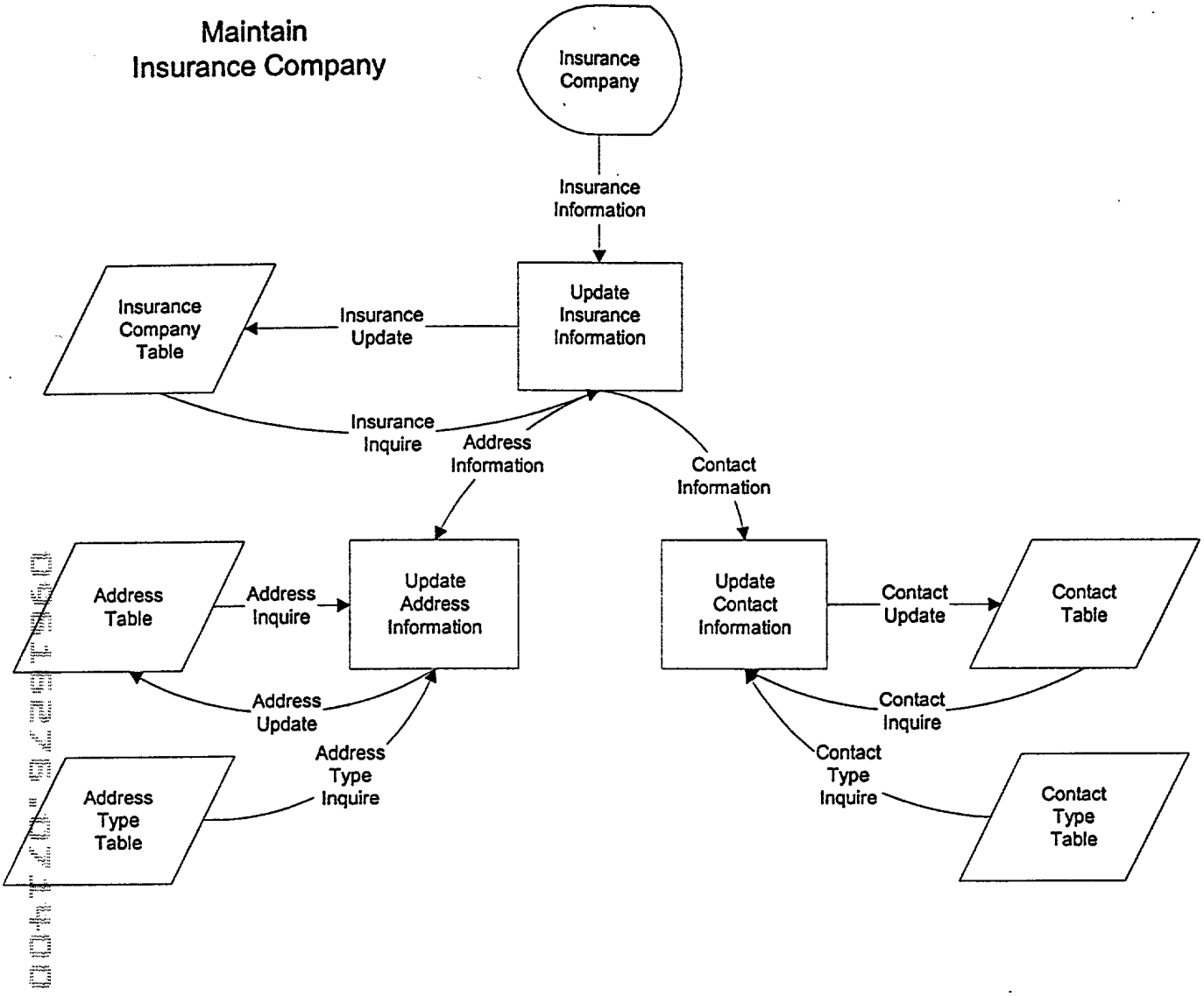

Doctor
Pharmaceutical Company
Hospital
Insurance Company
Pharmacy
Payment
Payment
Payment
Payment
Payment
Post Payments
Payment Inquire
Payment Update
Payment Received Table
Acct Reeceivable Update
Acct Receivable Table

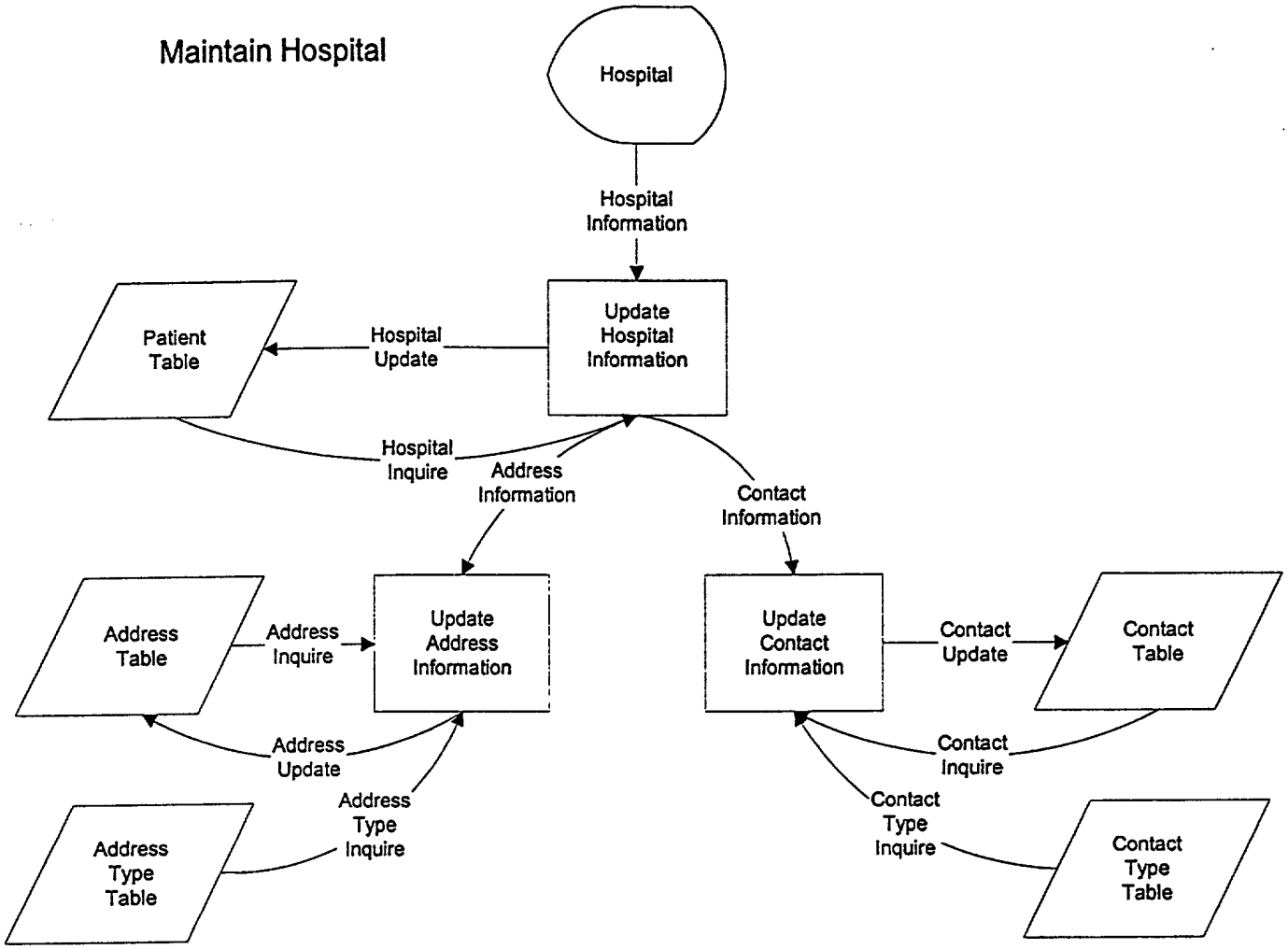

Financial Officer
Billing Request
Total Account Receivables
Account Recievable Sum
Total Account Payable
Account Payable Sum
Create Bills/ Checks
Bill Update
Check Inquire
Bill Table
Bill Inquire
Check Update
Created Bill/ Check
Check Table
Bill Inquire
Bill Update
Update Batch Number
Check Inquire
Check Update

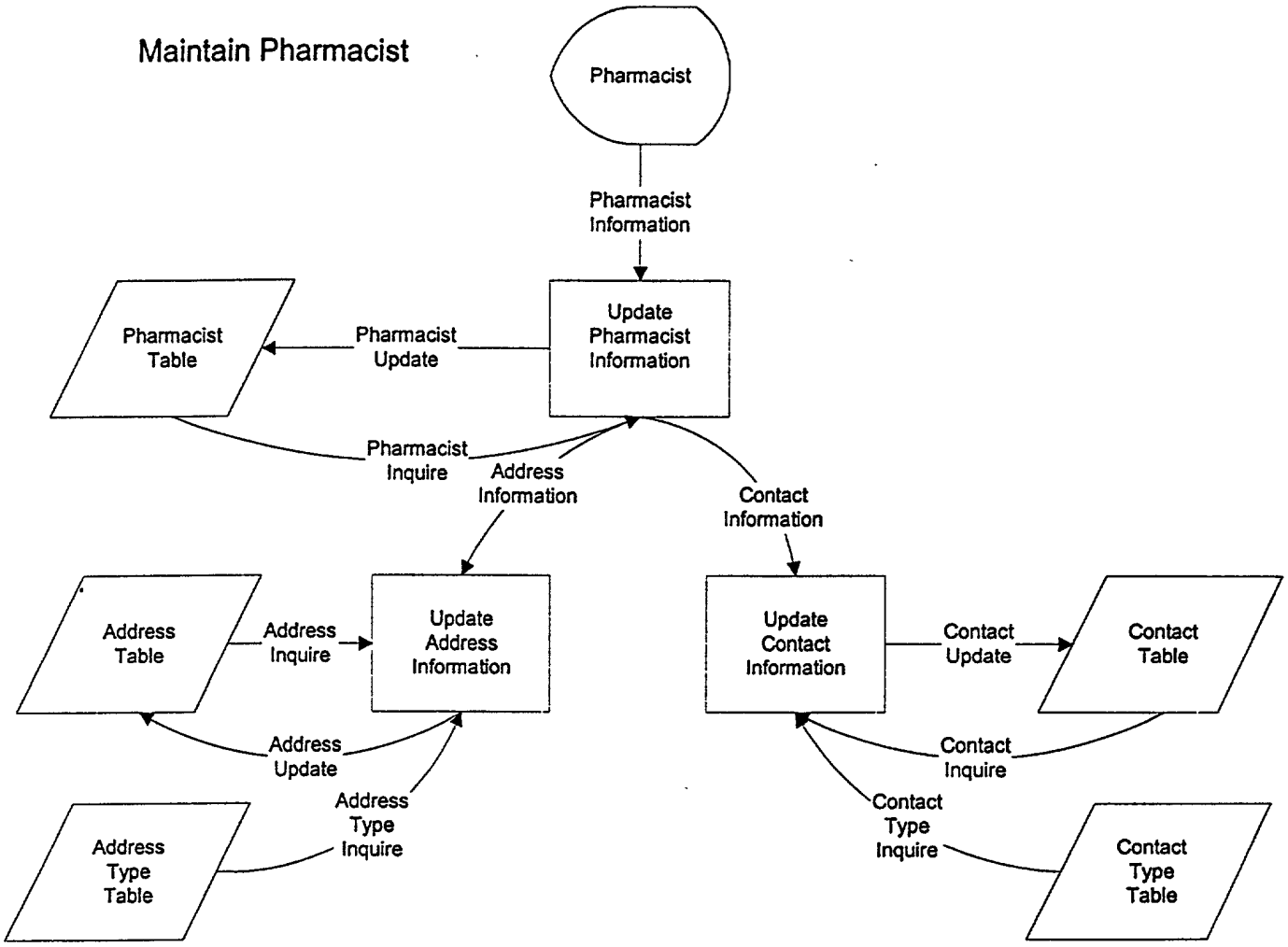
Maintain Doctor

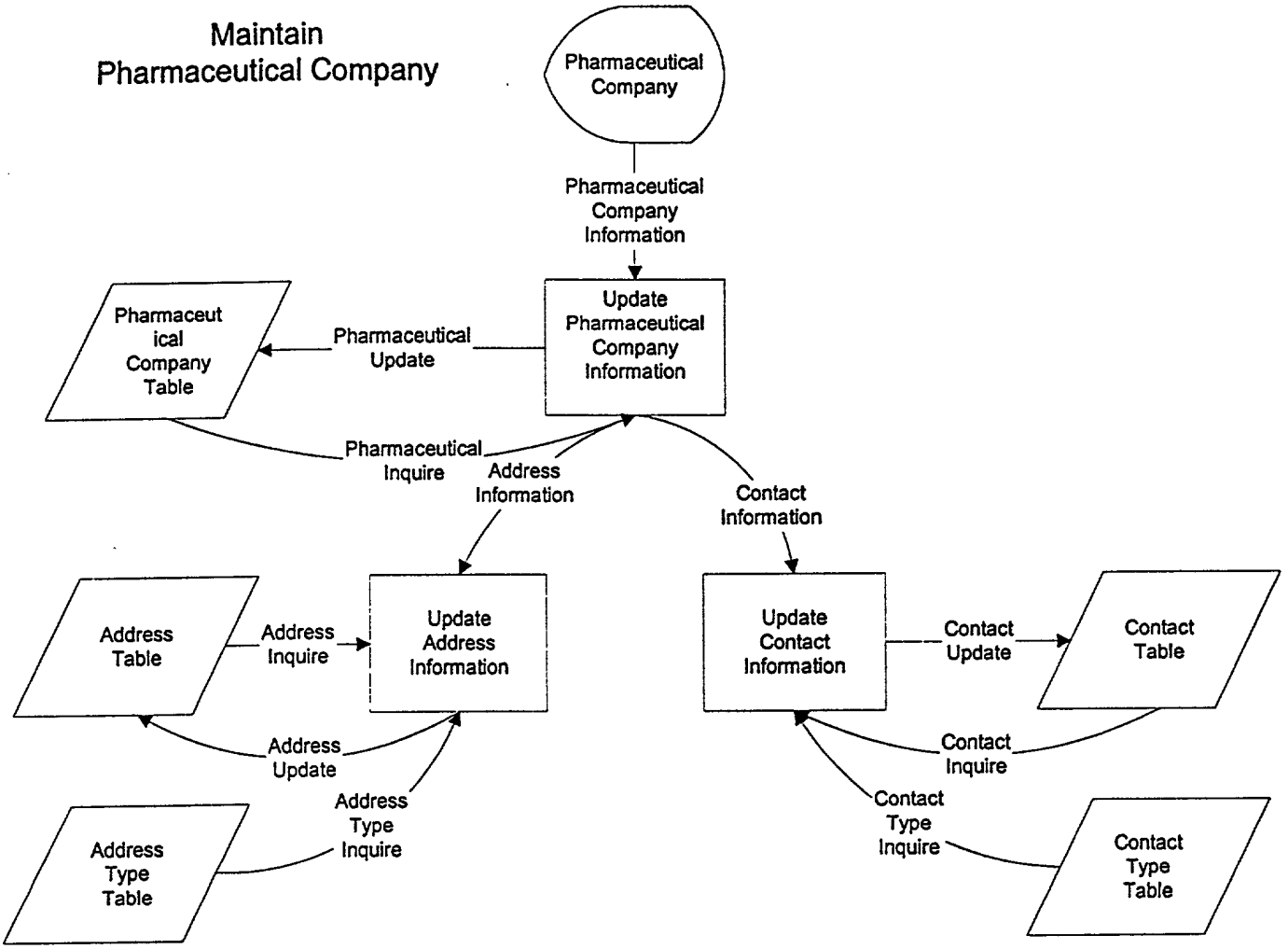





Maintain
Insurance Company
Insurance Company
Insurance Information
Update Insurance Information
Insurance Company Table
Insurance Update
Insurance Inquire
Address Information
Contact Information
Address Table
Address Inquire
Update Address Information
Update Contact Information
Contact Update
Contact Table
Address Update
Contact Inquire
Address Type Inquire
Contact Type Inquire
Address Type Table
Contact Type Table

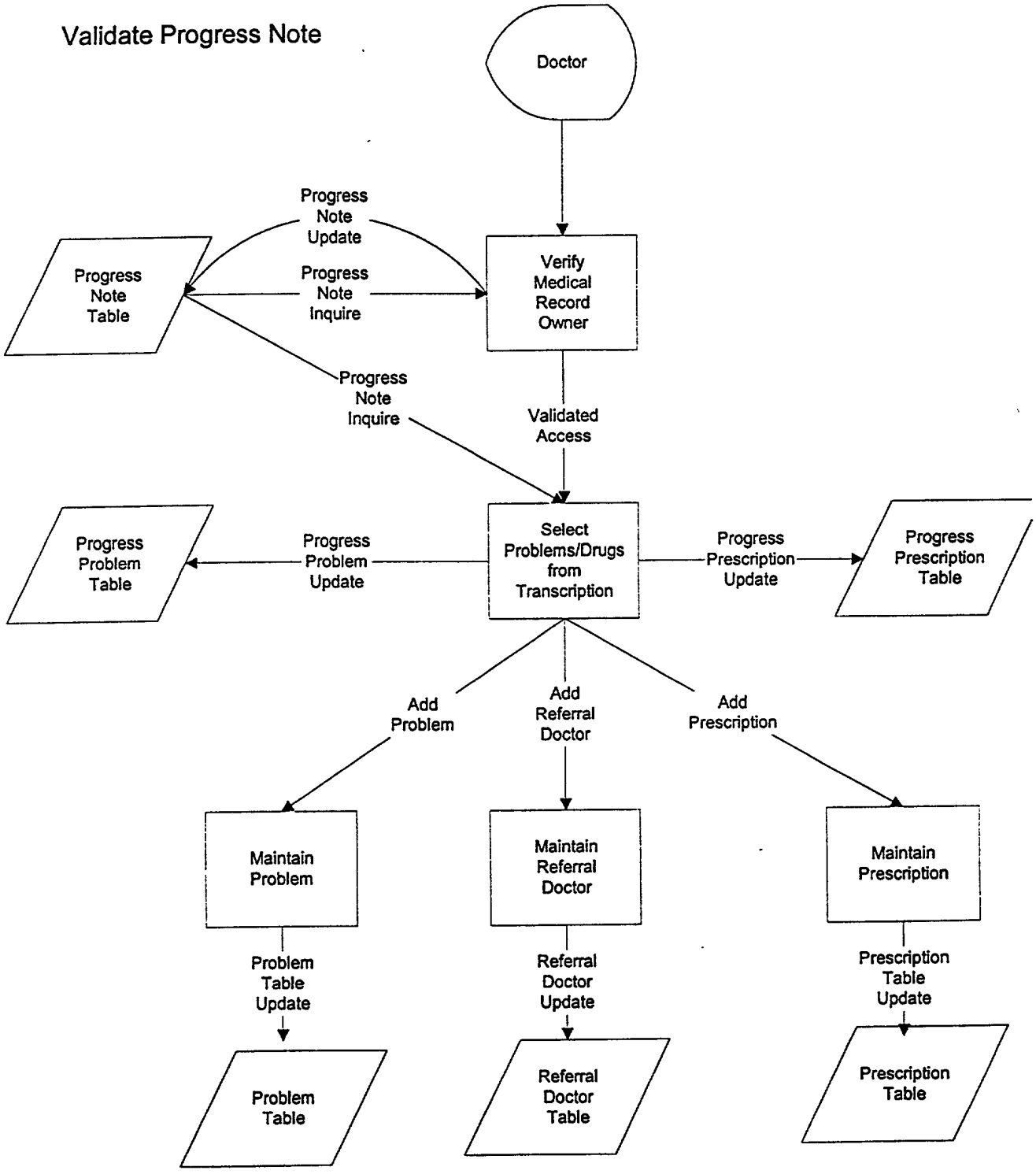




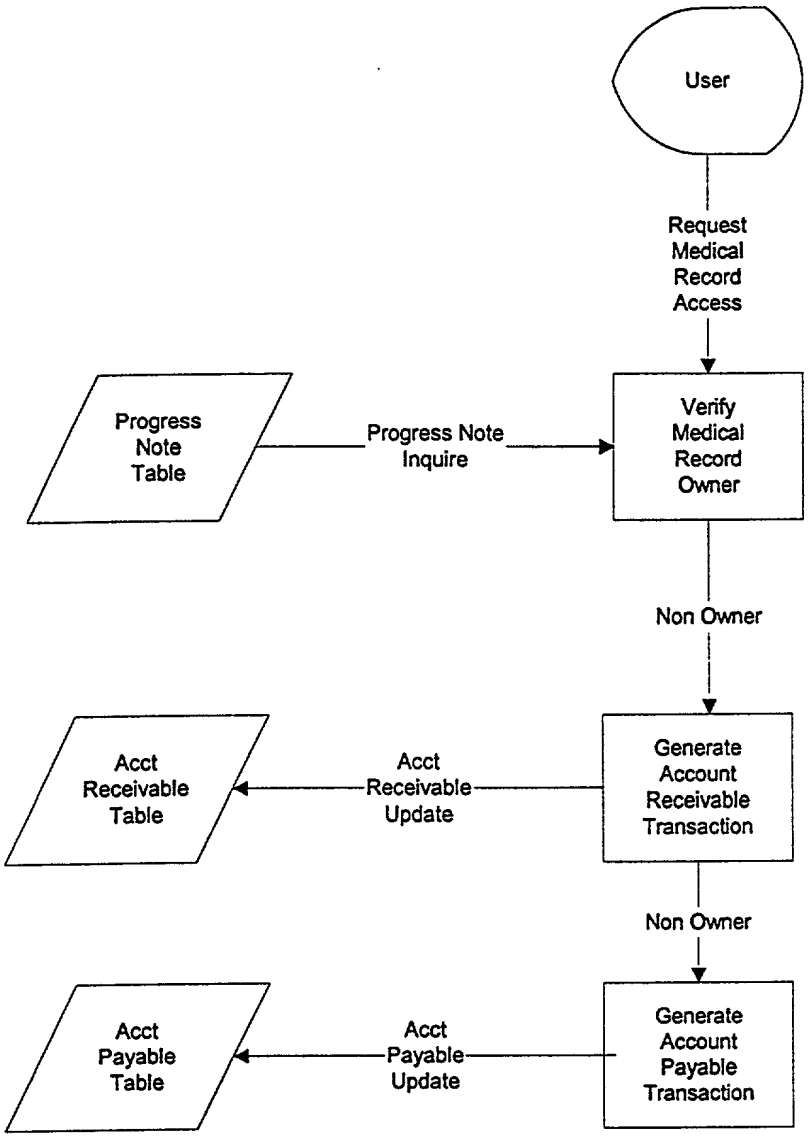

Maintain
Pharmaceutical Company
Pharmaceutical Company
Pharmaceutical Company Information
Update Pharmaceutical Company Information
Pharmaceut ical Company Table
Pharmaceutical Update
Pharmaceutical Inquire
Address Information
Contact Information
Address Table
Address Inquire
Update Address Information
Update Contact Information
Contact Update
Contact Table
Address Update
Address Type Inquire
Contact Inquire
Contact Type Inquire
Address Type Table
Contact Type Table

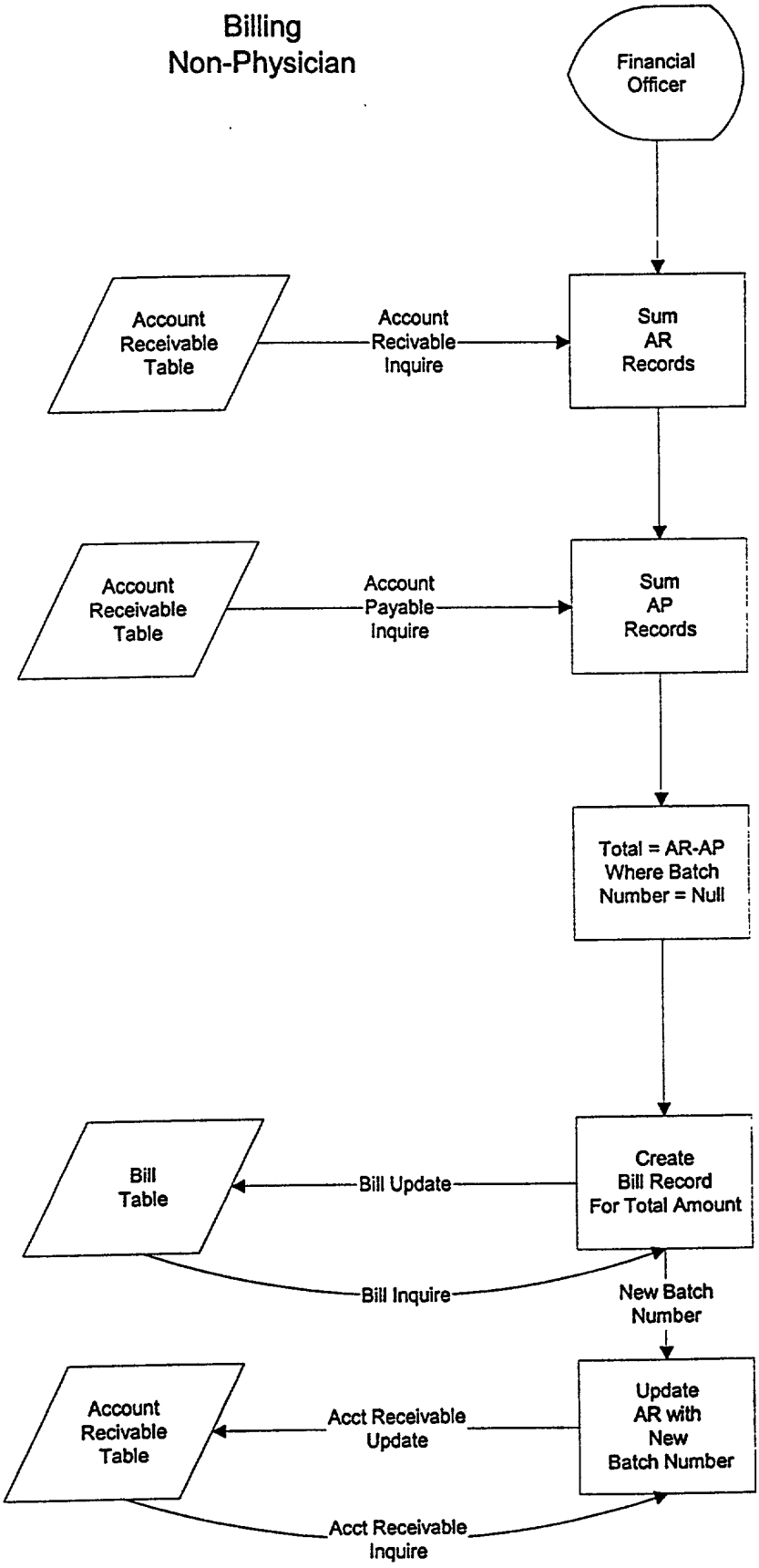
MEDISYN

Validate Progress Note

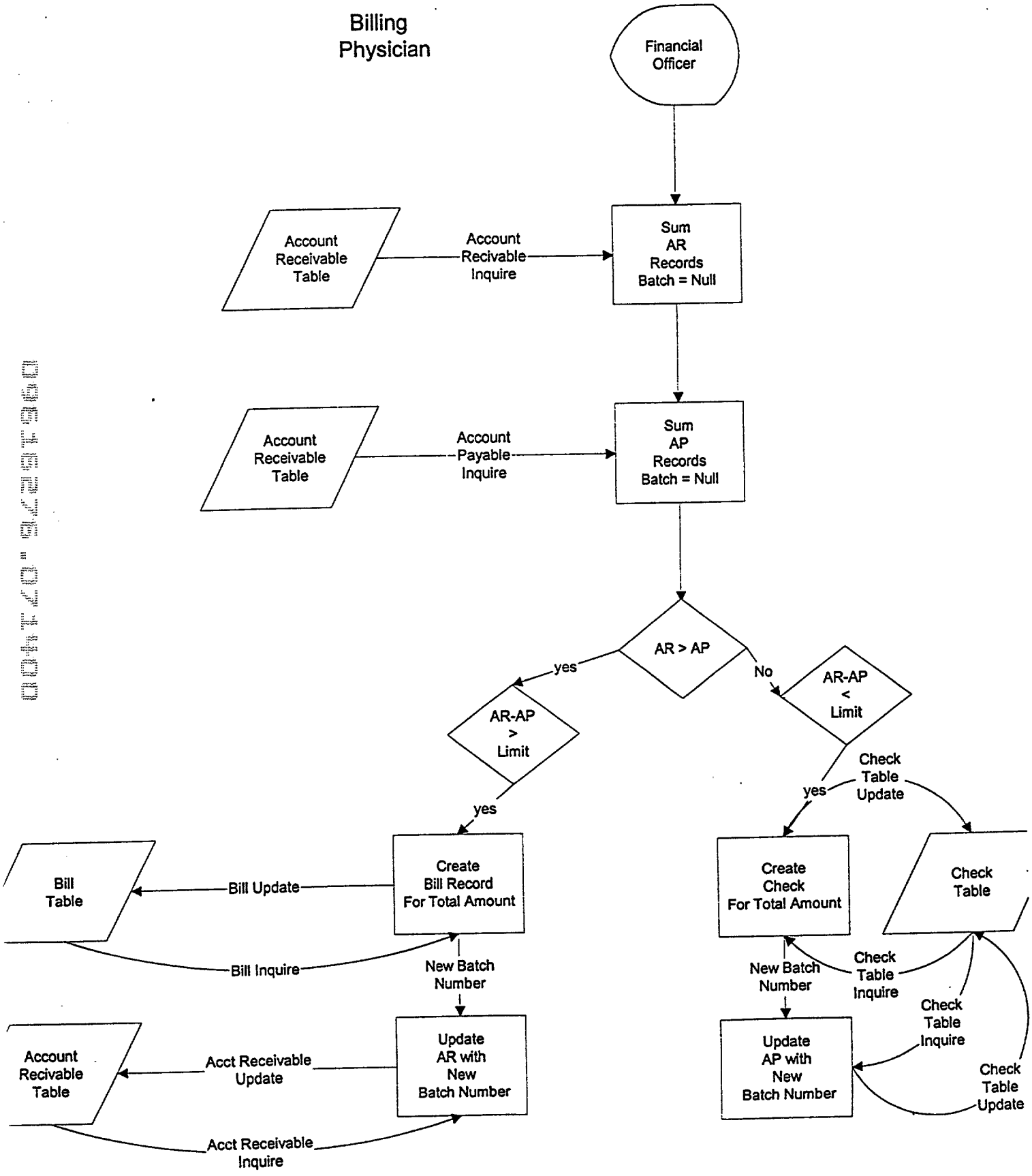


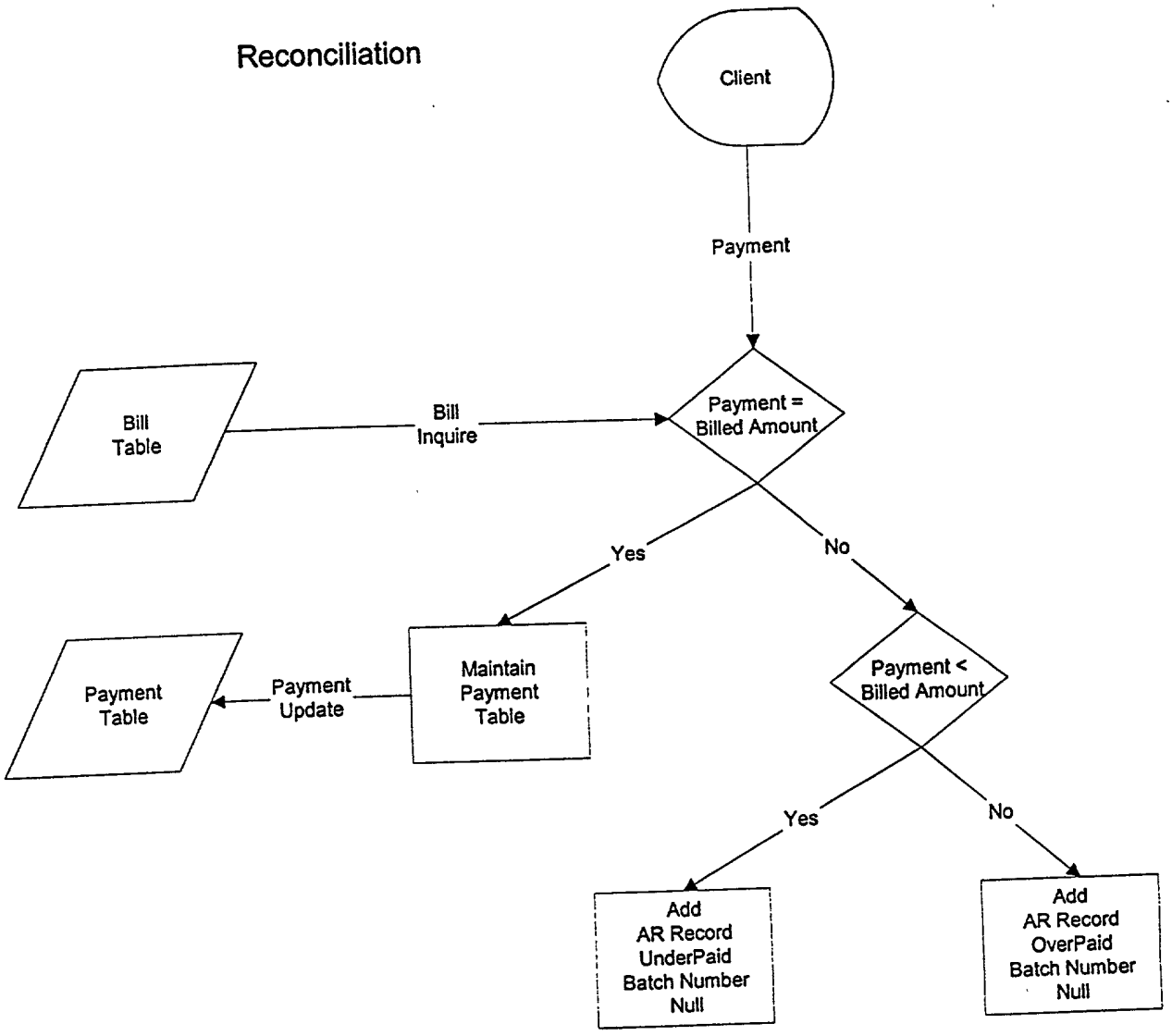
Record Access



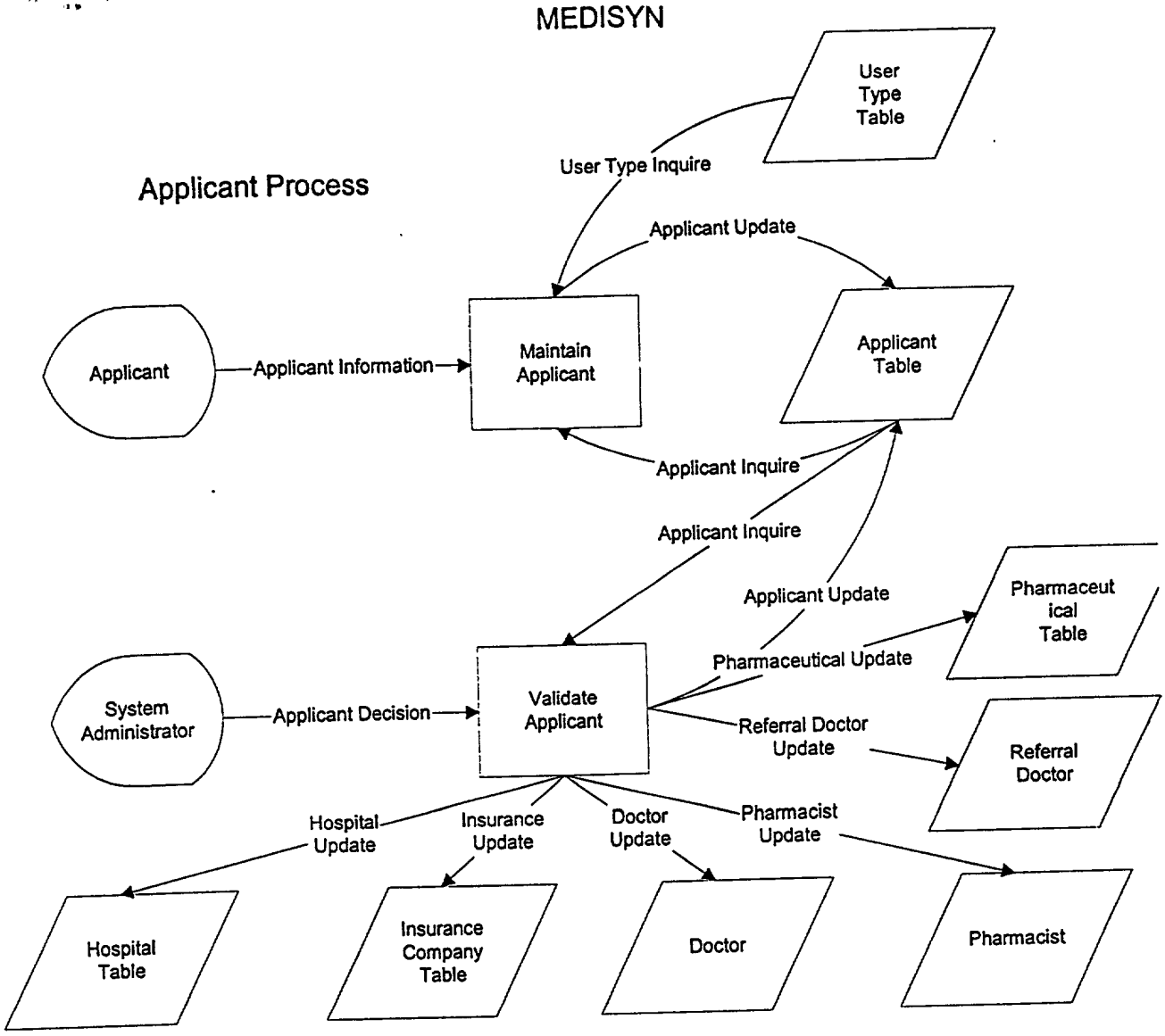


MEDISYN


Billing Physician
Financial Officer
Account Receivable Table
Account Recivable Inquire
Sum AR Records Batch = Null
Account Receivable Table
Account Payable Inquire
Sum AP Records Batch = Null
AR > AP
yes
No
AR-AP > Limit
AR-AP < Limit
yes
Check Table Update
Bill Table
Bill Update
Create Bill Record For Total Amount
Bill Inquire
New Batch Number
Create Check For Total Amount
Check Table
New Batch Number
Check Table Inquire
Check Table Inquire
Account Recivable Table
Acct Receivable Update
Update AR with New Batch Number
Update AP with New Batch Number
Check Table Update
Acct Receivable Inquire



MEDISYN


Applicant Process
User Type Table
User Type Inquire
Applicant Update
Applicant
Applicant Information
Maintain Applicant
Applicant Table
Applicant Inquire
Applicant Inquire
Applicant Update
Pharmaceutical Table
Pharmaceutical Update
System Administrator
Applicant Decision
Validate Applicant
Referral Doctor Update
Referral Doctor
Hospital Update
Insurance Update
Doctor Update
Pharmacist Update
Hospital Table
Insurance Company Table
Doctor
Pharmacist